UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

DEPARTAMENTO DE MATEMÁTICA APLICADA A LAS TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES

Técnicas criptográficas utilizadas en “malware”

Trabajo fin de grado, julio 2015

Eduardo Ruiz Azofra

UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior de Ingenieros de Telecomunicación

Trabajo fin de grado, julio 2015

Título: TÉCNICAS CRIPTOGRÁFICAS UTILIZADAS EN "MALWARE"

Departamento de Matemática Aplicada a las Tecnologías de la Información y las Comunicaciones

Autor: Ruiz Azofra, Eduardo

Tutor: Braquehais Lorente de No, Fernando

Miembros del tribunal:

- Presidente: Sánchez Ávila, Carmen
- Vocal: Villagrá Gonzalez, Víctor Abraham
- Secretario: Martín García, Lorenzo Javier
- Suplente: Ballesteros Olmo, Francisco

Fecha de lectura y defensa:

Calificación:

RESUMEN

La finalidad de este trabajo es estudiar los tipos de malware existentes así como las técnicas criptográficas utilizadas en ellos para ocultar y ofuscar sus actividades, con el fin de impedir su análisis a las empresas de seguridad. En este trabajo se propone un criptosistema conveniente que, además de en malware, pueda ser empleado en otros ámbitos.

El análisis comienza proponiendo una breve definición del término C*riptografía* y explicando mediante ejemplos los distintos tipos de sistemas criptográficos y algunos posibles usos.

A continuación se lleva a cabo un estudio de los distintos tipos de malware existentes, dando una visión histórica de los tipos de Criptografía utilizados hasta el momento en cada caso. El estudio profundiza en las técnicas criptográficas utilizadas actualmente y su seguridad criptográfica. Además, se realiza una propuesta de posibles soluciones criptográficas, analizando la seguridad computacional resultante de aplicar dichas soluciones y calculando la carga computacional adicional precisa. Finalmente se ofrecen una serie de conclusiones y unas líneas de desarrollo futuras.

Palabras clave: Criptografía, malware, criptosistema, seguridad criptográfica.

ABSTRACT

The purpose of this paper is to analyze the existing malware types and their corresponding cryptographic techniques currently. These techniques are used to hide their activities from security companies. In this paper we propose a convenient cryptosystem, which can be used in malware and other areas.

To begin this analysis with, a brief definition of Cryptography is proposed. Some examples are given to explain the different kinds of cryptographic systems and their possibilities of use.

In order to analyze cryptographic techniques used in malware, we study the existing malware types first. Afterwards, a historical overview of the cryptographic types is given, before studying deeper about this topic nowadays. In addition, we propose possible cryptographic solutions based on a computational security analysis resulting from implementing these solutions and calculating the additional computational load. Finally, some conclusions and recommendations for future development are offered.

Keywords: cryptography, malware, cryptosystem, cryptographic security.

ÍNDICE

ÍNDICE DE FIGURAS

Listado de acrónimos no universales

(Por orden de aparición)

PUP: Programa potencialmente no deseado.

CPC: Coste por clic.

CPM: Coste por impresión.

PPC: Pago por clic.

DoS: Ataque de denegación de servicios.

DDoS: Ataque de denegación de servicios distribuida.

RAT: Herramientas remotas de administración de sistemas.

ATM: Máquina utilizada para extraer dinero de una cuenta bancaria.

MBR: Registro de arranque del disco.

TEA: *Tiny Encryption Algorithm*.

C&C: Panel de control.

CBC: Cifrado encadenado de bloques.

ECDH: *Elliptic curve Diffie–Hellman*.

PoS: *Points of Sale*. También llamado *TPV* (terminal punto de venta).

Blob: *Binary large object*.

SID: ID de sección.

LCG: *Linear Congruential Generator*.

EVFS: Sistema virtual de archivos cifrados.

DES: *Data Encryption Standard*.

AES: *Advanced Encryption Standard*.

CPA: Ataque de texto elegido.

KPA: Ataque de texto conocido.

1. OBJETIVOS Y ESTRUCTURA

La finalidad de este trabajo es estudiar los tipos de malware existentes así como las técnicas criptográficas utilizadas en ellos para ocultar y ofuscar sus actividades, con el fin de impedir su análisis a las empresas de seguridad. En este trabajo se propone un criptosistema conveniente que pueda ser empleado en otros ámbitos, además de en malware.

1.1. Objetivos

La secuencia de objetivos que se pretenden conseguir en este trabajo son los siguientes:

1. *Definir y entender la Criptografía*: se estudia el término de Criptografía y los distintos tipos en los que se clasifica. Además se explican las distintas finalidades de su uso y cómo se pueden lograr esas finalidades.

2. *Definir y clasificar los tipos de malware*: se define el amplio término *malware* diferenciando sus distintos tipos, así como los distintos efectos que pueden tener en la actualidad.

3. *Analizar los distintos criptosistemas utilizados en el ámbito del malware*: se investiga la Criptografía utilizada en malware a lo largo de la historia, además de analizar en mayor profundidad los criptosistemas utilizados en malware en la actualidad.

4. *Analizar la seguridad criptográfica del malware*: se definen los distintos tipos de seguridad criptográfica existentes y se analiza la seguridad de los criptosistemas encontrados en malware.

5. *Proponer un criptosistema adecuado*: se propone el criptosistema más adecuado para las actividades realizadas por malware, que también puede ser utilizado en otras aplicaciones. Además se analiza el impacto en la carga computacional que puede ocasionar la adaptación de este criptosistema.

1.2. Estructura

El documento se estructura en tres partes para conseguir estos objetivos.

La primera parte del documento introduce los temas que serán desarrollados. De esta forma, se propone una breve definición del término *Criptografía* explicando mediante ejemplos los distintos tipos de sistemas criptográficos y algunos posibles usos. Para completar este punto, se describe brevemente el malware.

La segunda parte comienza con una descripción de los tipos de malware conocidos. A continuación se da una visión histórica de la Criptografía en este malware antes de llevar a

cabo el análisis de los tipos de Criptografía empleados en el malware actual. También se analiza la seguridad criptográfica en estos casos.

Finalmente, se realiza una propuesta de posibles soluciones criptográficas que puede ser conveniente implementar en cada caso, así como un cálculo de la carga computacional adicional necesaria si se implementasen esas soluciones. Como conclusión, se hace un análisis del trabajo realizado además de ofrecer unas líneas de desarrollo futuras.

2. INTRODUCCIÓN

Se entiende actualmente la Criptografía como la ciencia que hace uso de métodos y herramientas matemáticas con el objetivo principal de proteger la información por medio de un algoritmo de cifra.

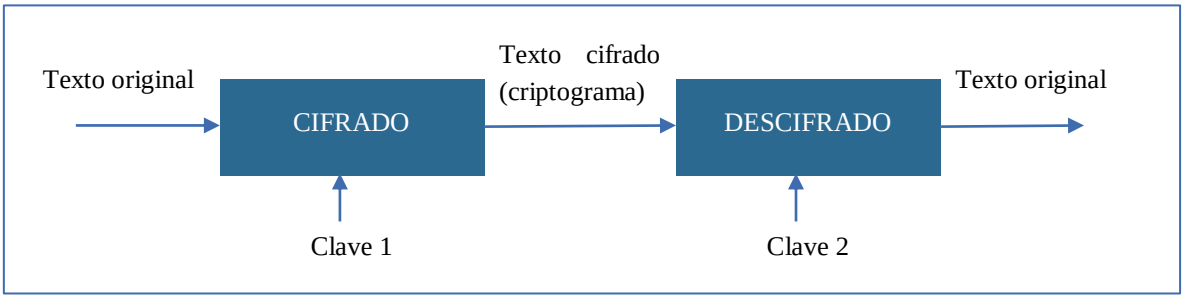

Figura 1. Proceso de cifrado/descifrado

En la Figura 1 se muestra el esquema general de un proceso de cifrado/descifrado. La clave utilizada en el proceso de descifrado puede ser o no igual a la del proceso de cifrado. Según esto, los procesos de cifrado/descifrado se pueden clasificar según el siguiente criterio:

- *Clave simétrica*: las claves 1 y 2 son iguales. El proceso se llama en este caso cifrado simétrico o cifrado de clave secreta.

- *Clave asimétrica*: en caso contrario, las claves 1 y 2 son distintas y el proceso es un cifrado asimétrico o cifrado con clave pública.

En la Criptografía clásica, el cifrado se basaba en la transposición o permutación de caracteres mediante un sistema acordado previamente entre el emisor y el receptor. Así, en el cifrado César, se mueve cada letra del abecedario un número determinado de posiciones. Por ejemplo, moviendo las letras cuatro posiciones se corresponderán la letra "a" a la "d", la "b" a la "e", etc. Un ejemplo de Criptografía simétrica es la máquina Enigma, muy utilizada en la década de 1920. Enigma era una máquina de cifrado electromagnética que generaba abecedarios según la posición de unos rodillos que podían tener distintas órdenes y posiciones [28].

Los sistemas de cifrado de la Criptografía clásica, así como algunos pertenecientes a la Criptografía moderna (a partir de la segunda mitad del s. XX), han utilizado cifrado de clave simétrica. La publicación de C. Shannon de sus estudios en teoría de la comunicación en 1948, unido al nacimiento de la computación, provocó que la información se empezase a representar por medio de conjuntos de bits, no de caracteres, con lo que se empezó a cifrar esos bits [71].

En la Criptografía moderna se pueden clasificar los sistemas de cifrado simétricos en dos grandes grupos: cifrado en flujo, en el que se cifra la información bit a bit, y cifrado en

bloque, en el que se cifran bloques de varios bits. Estos sistemas son, en general, más rápidos y eficaces que los de clave simétrica pero tienen un problema a la hora de distribuir las claves. El canal para esta distribución debe ser seguro.

Los sistemas de clave asimétrica por propia definición no tienen el problema de distribución de claves. Estos sistemas no permiten, a partir de la clave pública (la clave 1 en la Figura 1, es decir, la que permite cifrar), calcular la clave privada (clave 2 en la Figura 1). La clave privada de un sujeto X se denomina en adelante K_{prX} y la pública se denomina K_{puX}. Gracias a esto, cualquier usuario conoce la clave pública y puede cifrar información para el poseedor de la clave privada. Por ejemplo, un emisor Alice podrá cifrar información con la clave pública de Bob (K_{puB}), de forma que únicamente Bob pueda acceder a esa información con su clave privada (K_{prB}).

En la actualidad, la Criptografía se puede utilizar con tres finalidades independientes o conjuntas:

- *Confidencialidad*: la información está accesible únicamente para usuarios autorizados.
- *Autenticidad*: garantiza la identidad del remitente y del destinatario de la información. De esta forma se asegura la característica del *no-repudio*, que previene la posibilidad de que un emisor niegue haber remitido un mensaje (cuando realmente lo haya emitido) y que un receptor niegue haberlo recibido (cuando realmente lo haya recibido).
- *Integridad*: garantiza que la información no ha sido manipulada ni alterada. La manipulación de parte del mensaje cifrado o de su totalidad por parte de un agente externo que lo intercepte provoca que el destinatario, en principio, no pueda descifrarlo [64].

Continuando el anterior ejemplo, se puede observar que la autenticidad del destinatario del mensaje está garantizada, puesto que Bob es el único que puede descifrar el mensaje. En este caso, sin embargo, no están garantizadas la autenticidad del emisor ni la integridad del mensaje puesto que la clave de cifrado (K_{puB}) es pública.

Para solucionar estos problemas se puede recurrir a la firma digital. El emisor cifra el mensaje con su clave privada (K_{prA}) de forma que cualquiera que posea su clave pública (K_{puA}) puede descifrarlo, comprobando así la autenticidad del remitente. Además, la integridad se verifica porque el mensaje enviado por Alice es la suma del mensaje original con el mensaje cifrado con la clave K_{prA}. De esta forma, Bob puede descifrar el mensaje cifrado y compararlo con el original. Para llevar esto a cabo se siguen los siguientes pasos:

- Alice firma el mensaje con su clave privada (K_{prA}).
- Alice junta el mensaje firmado y el original y los cifra con la clave pública de Bob (K_{puB}).

- Bob recibe el mensaje de Alice y lo descifra con su clave secreta (K_{prB}).
- Bob separa el mensaje firmado y lo descifra con la clave pública de Alice (K_{puA}). A continuación, compara el mensaje descifrado con el original comprobando así la autenticidad del emisor y la integridad del mensaje.

Para firmar un mensaje largo es necesario utilizar una clave igualmente larga, por ello es conveniente acortar el mensaje firmado. Una forma de hacerlo es aplicar una función *hash* (*algoritmo de resumen* en castellano, representado por la función H(m)) obteniendo una *huella* o *resumen* del mensaje original [27]. Un *hash* es una función unidireccional que convierte una cadena de caracteres de cualquier longitud en un conjunto de caracteres de longitud fija.

La función *hash* tiene una serie de propiedades (bajo costo, compresión, rango variable, función uniforme) que la hacen muy útil en el ejemplo de Alice y Bob. Así, Alice puede aplicar una función *hash* conocida por el receptor (en general, la función *hash* será pública) al mensaje original, reduciendo así la longitud del mensaje y obteniendo H(m). Alice cifra el *resumen* con su clave privada (K_{prA}) y lo junta, como en el ejemplo anterior, al mensaje original. Después, cifra el conjunto con la clave pública de Bob (K_{puB}). Una vez descifrado el conjunto con su clave privada (K_{prB}), Bob descifra el *resumen* cifrado con la clave pública de Alice (K_{puA}). Finalmente, Bob aplica la función hash al mensaje original y lo compara con el *resumen* descifrado.

Los métodos de cifra y firma digital son utilizados, por ejemplo, en la distribución de software, en el envío de correos electrónicos y en el acceso a diferentes plataformas web. No es sorprendente que estos métodos sean utilizados también por quienes quieren ocultar contenidos o realizar actividades ilegales. Un posible caso de actividad ilegal puede ser la distribución y el uso de *malware*.

Un código malicioso o *malware* es cualquier tipo de programa desarrollado para causar daños o introducirse de forma no autorizada en algún sistema informático [27]. Existen muchos tipos de *malware* como virus, *adware*, *backdoors*, gusanos, etc.

En la actualidad, las principales vías de infección de *malware* en los dispositivos (ordenadores, *tablets* y *smartphones*) son programas gratuitos infectados, sitios web fraudulentos, redes sociales, dispositivos externos USB o discos, archivos adjuntos en correos electrónicos no solicitados. En los últimos años, el número de códigos maliciosos ha crecido exponencialmente. Vemos las diferentes tipologías en el siguiente capítulo.

3. TIPOS DE MALWARE

Malware es un término muy amplio que engloba numerosos códigos maliciosos y otros programas no deseados que pueden clasificarse siguiendo distintos criterios. El criterio seguido en la siguiente clasificación es la forma de propagación o infección. Otro posible criterio de clasificación es el dispositivo objetivo del malware (dispositivo móvil, ordenador) o, más específico, el sistema operativo objetivo (Windows, Android, Mac OS, iOS, etc). Según este último criterio, por ejemplo, los dos sistemas operativos que son objetivo de una mayor proporción de malware son los dos primeros (Windows y Android) en sus diferentes versiones.

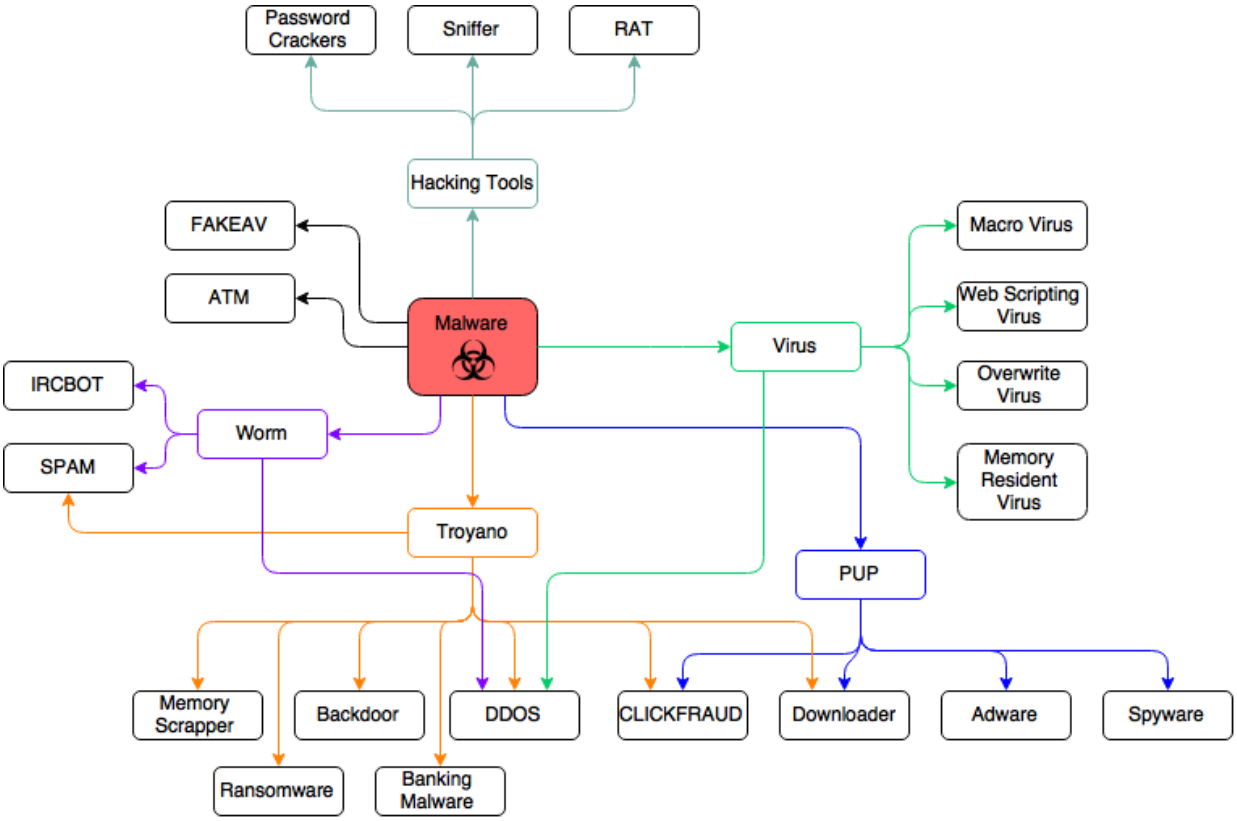

Figura 2. Tipos de malware

Al clasificar los tipos de malware por su forma de propagación o infección se obtienen las siguientes categorías [76]. Estas pueden clasificarse también en distintos tipos según la finalidad del código malicioso. Algunos de estos tipos de malware pueden pertenecer a varias categorías.

3.1. Trojan Horse (Troyano)

El *trojan horse*, también llamado *trojan* (*troyano* en castellano), es software malicioso aparentemente legítimo e inofensivo: por ejemplo un archivo con nombre informe.pdf.exe. Dependiendo de la configuración del ordenador y del sistema operativo, las extensiones de los archivos (*.exe* en este caso) no se muestran, por lo que el usuario ve

un archivo informe.pdf y, creyendo que es realmente un informe, lo abre/ejecuta infectando el sistema. Este tipo de malware se propaga mediante la instalación de los archivos "inofensivos" nombrados anteriormente. Los troyanos pueden clasificarse por su finalidad u objetivos en:

3.1.1. Ransomware

Este tipo de malware mantiene cautivo el dispositivo exigiendo un rescate. El malware cifra archivos del disco o bloquea el acceso al sistema hasta que el usuario paga al creador del malware.

3.1.2. Downloader

El downloader por sí sólo no es código malicioso, sino software diseñado para bajar otros archivos. Es utilizado comúnmente para descargar malware de forma inadvertida, ya que el software de seguridad como antivirus no lo identifican como malicioso. Esto se debe a que en el propio sistema operativo o en programas legítimos de terceros existe código similar para, por ejemplo, la descarga e instalación de actualizaciones de esos programas. Puede considerarse también dentro de la categoría de PUP, explicada más adelante.

3.1.3. Click-fraud

Los troyanos de tipo click-fraud están diseñados para generar clicks no válidos en enlaces de publicidad para hacer subir el precio de la publicidad al anunciante. De esta forma es posible incluso arruinar a una pequeña empresa con presupuesto limitado para marketing. Si esta empresa tiene que pagar un coste por click (CPC) o por impresión (CPM), puede salirle muy caro que alguien realice miles de clicks en sus anuncios sin realmente comprar nada. También es posible sacar provecho económico mediante click-fraud. Si el propietario de una página web consigue que se produzcan clicks en los anuncios de su propia página ganará dinero por cada click realizado en un anuncio, si el modelo de marketing utilizado es pay-per-click (PPC). El atacante intentaría controlar el mayor número de nodos posible para realizar de forma automática visitas a sus páginas web con contenidos publicitarios y acceder a los banners de publicidad [30].

3.1.4. DDOS

Un troyano de tipo DDOS es aquel que convierte al sistema en un *bot* para realizar ataques de denegación de servicios (DoS o *Denial of Service*, en inglés). El *bot master*, el que controla estos bots, puede ordenar a los sistemas controlados que actualicen el propio troyano, que descarguen otro malware o que realicen ataques DoS. Si este ataque se realiza simultáneamente desde distintos equipos de la *botnet*, que es un conjunto de bots, se denomina DDoS o ataque de Denegación de servicios distribuida (*Distributed DoS*) [50]. Estos ataques DDoS pueden utilizarse para, por ejemplo, saturar servidores hasta que

tengan que ser reiniciados. Un ejemplo conocido son los ataques DDoS de la sociedad Anonymous a páginas del gobierno estadounidense.

3.1.5. Backdoor

El backdoor es un tipo de malware diseñado para acceder al dispositivo eludiendo los mecanismos de seguridad del dispositivo aprovechando puertas traseras o backdoors instaladas por el programador. Este puede querer acceder al programa para solucionar problemas o realizar actualizaciones automáticamente. Los atacantes pueden explotar utilizar estas backdoors como parte de un *exploit*. Un ejemplo de troyano de tipo backdoor es Regin. Regin realiza un ataque en varias etapas aprovechando puertas traseras con el fin de monitorizar la actividad del usuario y robar datos. Este troyano en concreto tiene una estructura modular (ver punto *5.6 Otros*).

3.1.6. Memory Scrapper

El memory scrapper es un tipo de troyano que ayuda al atacante a encontrar información personal en el sistema infectado. Examina la memoria del sistema buscando datos no disponibles a través de otros procesos.

3.1.7. Banking Malware

Este tipo de troyanos, llamados también *troyanos bancarios*, están diseñados específicamente para entrar en una cuenta bancaria online y transferir dinero a otras cuentas controladas por los atacantes. Una vez infectado el sistema con uno de estos troyanos, este espera inactivo hasta que el usuario entra en la página web de su banco. Entonces el troyano roba la información de acceso online a la cuenta sin que el usuario se percate y la envía al atacante. Este entra en la cuenta bancaria y transfiere el dinero a otra cuenta, normalmente a una *money mule* (cuentas bancarias de estudiantes de intercambio, por ejemplo). Al cabo de unos días retiran el dinero de esas *mulas*. Algunos troyanos bancarios hacen las transferencias internacionales a las cuentas de los atacantes directamente [73]. El ejemplo de troyano bancario más conocido es Zeus, cuyas distintas variantes han infectado a lo largo de los años más de 13 millones de ordenadores y numerosos dispositivos móviles Android y Blackberry.

3.2. PUP

Los programas potencialmente no deseados (PUP o *Potentially Unwanted Programs*, en inglés) son programas que se instalan en el sistema sin que se notifique al usuario de la instalación (o con avisos que el usuario medio puede no saber interpretar correctamente). Generalmente, el usuario no querría realizar la instalación de estos programas de saber sus posibles finalidades. Los PUP son capaces de cambiar la configuración del ordenador, mostrar constantemente anuncios o modificar los resultados de las búsquedas en Internet.

La mayoría de los PUP son funcionalmente similares, si no iguales, a los troyanos [41]. Los tipos de PUP más conocidos son:

3.2.1. Adware

El adware es un tipo de PUP que muestra anuncios de forma automática. Su forma de propagación más común es la inclusión de adware al instalar software y aplicaciones en su versión "gratuita". Es común encontrar adware acompañado de spyware. Un ejemplo de adware es *Movies Toolbar* (puede ser encontrado también como *Music Toolbar* o *Settings Manager*). Este programa se instala junto a otros programas de descargas libres y, una vez el ordenador lo tiene instalado, la página de inicio del navegador y el motor de búsqueda se cambian por *search.ask.com* [49].

3.2.2. Spyware

El spyware es un tipo de malware que espía al usuario sin que este lo sepa. Puede ser capaz de monitorizar la actividad del usuario (pulsaciones del teclado, por ejemplo), recolectar datos como nombres de usuario y contraseñas o incluso datos bancarios. Además puede contar con la posibilidad de modificar ajustes de seguridad de navegadores u otros programas. Su propagación habitual se produce explotando vulnerabilidades de software o acompañando software legítimo o a un troyano.

3.3. Virus

Este tipo de malware es capaz de replicarse a sí mismo para propagarse a otros dispositivos adhiriéndose a diversos programas. El usuario, al ejecutar estos programas, ejecuta también el código del virus e infecta sin saberlo el sistema. Otras vías de propagación pueden ser *scripts* (archivos de comandos ejecutables), a través de vulnerabilidades utilizando *exploit-kits* en aplicaciones web o mediante archivos adjuntos en correos electrónicos. En cualquier caso, el virus únicamente puede replicarse y propagarse mediante una acción del usuario. Por ejemplo, algunos virus se propagan infectando memorias USB. Estas memorias son infectadas al conectarlas a una máquina infectada pero para propagarse es necesario que las conecten en otras máquinas. Hace años, antes de las memorias USB podían utilizarse, por ejemplo, *floppy disks* (*disquetes* en castellano).

La finalidad principal de los virus es infectar al mayor número de máquinas posible. Como componente adicional pueden robar información, dañar el sistema infectado o crear botnets con los equipos infectados. Los tipos principales de virus son [75]:

3.3.1. Macro Virus

Reemplaza las *macros* de los ficheros (pdf, doc, etc) por sus propias macros. "Una macro es un conjunto de comandos utilizados por los programas para realizar acciones

habituales como abrir documentos" [33]. Al cambiar, por ejemplo, la macro de abrir un documento por su propia macro, el virus podrá ejecutarse cada vez que se abra el documento.

3.3.2. Memory Resident Virus

El virus se almacena en la memoria del sistema y son activados al iniciar el sistema operativo. De esta forma, se esconden en la memoria RAM y son capaces de infectar archivos abiertos.

3.3.3. Overwrite Virus

Como indica su nombre, sobrescriben la información de los ficheros que infectan dejándolos inservibles. Al sobrescribir la información, el fichero permanece del mismo tamaño anterior, ocultándose así el rastro.

3.3.4. Web Scripting Virus

Aprovechan vulnerabilidades cuando son ejecutados como parte de scripts cargados por aplicaciones web en un navegador (Chrome, Firefox, etc).

3.4. Worm (Gusano)

Este tipo de software malicioso se propaga explotando las vulnerabilidades de los sistemas operativos conectados a una red (Ethernet, WAN…). Comparte la propiedad del virus de replicarse a sí mismo aunque, a diferencia de este, no necesita ser ejecutado por el usuario sino que se propaga de forma automática. De esta forma, el gusano puede replicarse y propagarse de forma automática miles de veces por segundo, lo que en la mayoría de los casos provoca problemas en la red (consumiendo ancho de banda, por ejemplo). Otra diferencia frente al virus estriba en que el gusano se encuentra en memoria, no adherido a un programa. Se pueden encontrar distintos tipos de gusano clasificados por su comportamiento:

3.4.1. IRCBOT

El IRCBOT es un tipo de gusano que se conecta a un servidor remoto de Internet Relay Chat (IRC). De esta forma puede recibir y ejecuta órdenes de un atacante remoto. Un ejemplo de este malware es el *Backdoor:Win32/IRCbot.K*, que puede incluso actualizarse o participar en ataques DDoS si el atacante lo desea.

3.4.2. Spam

Spam es el nombre dado a correos electrónicos basura no solicitados. Existe spam inofensivo cuya finalidad es la publicidad. Estos correos electrónicos pueden contener gusanos y algunos de estos pueden propagarse de nuevo accediendo a los contactos del

correo electrónico y reenviando el spam. Este spam puede contener también enlaces para realizar, por ejemplo, *phising*.

3.5. Hacking Tools

Las *hacking tools* son software diseñado para asistir o ayudar a un *hacker*. Un hacker es un experto en ramas técnicas relacionadas con la informática (programación, redes, sistemas operativos) [78]. Este software puede considerarse una rama de malware cuando es utilizada por hackers criminales (*crackers*). Algunas hacking tools son:

3.5.1. Sniffers

Herramienta que monitoriza el tráfico de una red. Puede ser utilizado de forma legítima en la administración de una red pero también es usado por crackers para robar información de la red. Esta herramienta puede consistir en software o hardware [43].

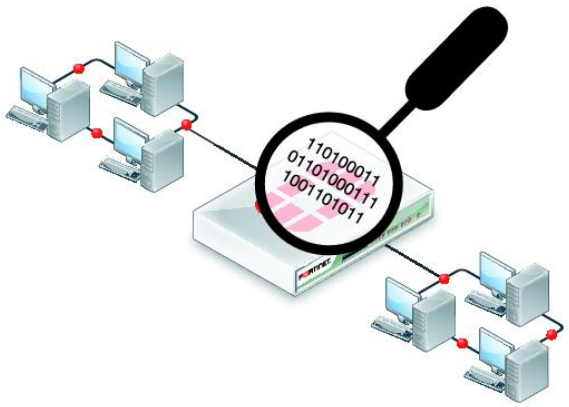

Figura 3. Sniffer

3.5.2. Remote Administration Tools (RAT)

Las herramientas de administración remotas permiten a un usuario controlar otro sistema a distancia como si estuviese físicamente utilizando esa máquina. Existen algunos RAT de uso comercial como *Remote Server Administration Tool for Windows 7* [81]. A pesar de ello, el término RAT suele asociarse a herramientas maliciosas que aprovechan el acceso remoto de una máquina para acceder, por ejemplo, a la información tomada por la webcam del ordenador sin que el usuario se percate de ello [44].

3.5.3. Password Crackers

Software diseñado para permitir recuperar al usuario o administrador de un equipo contraseñas perdidas u olvidadas. En las manos de un atacante estas herramientas pueden ser una amenaza a la seguridad y privacidad del usuario.

3.6. Fake AV

El Fake AV (también llamado FakeAlert o Fake AntiVirus) es un tipo de código malicioso diseñado para producir falsos avisos de antivirus o escáneres en un intento de engañar al usuario para que crea que su sistema está infectado. La falsa alerta ofrece usualmente la posibilidad de limpiar el sistema a cambio de una cuota de registro. El

FakeAlert en la mayoría de los casos únicamente coge los datos bancarios sin hacer cambios en el sistema ni arreglar los daños causados.

3.7. ATM

Un ATM o *Automated Teller Machine* es una máquina expendedora utilizada para extraer dinero de una cuenta bancaria. Existe malware enfocado a dar control y acceso a cajeros electrónicos. Ploutus y NeoPocket son algunos ejemplos de este tipo de malware. En la Figura 4 se observa un esquema del tipo de ataque realizado por NeoPocket.

Figura 4. Ploutus

3.8. Funcionalidades malware

La mayoría del malware creado en la actualidad está compuesto de distintos módulos o piezas que añaden distintas características al conjunto. Existen algunas características comunes a varios tipos de malware. Las más importantes se describen a continuación:

3.8.1. Rootkit

Una característica común tanto en troyanos como en virus y gusanos es la funcionalidad rootkit. Esta funcionalidad permite ocultar ficheros, acceder a información almacenada en el sistema o modificar la configuración del software instalado, incluido software de seguridad sin ser detectado por el usuario ni por el software de seguridad que tenga instalado. De esta forma puede ocultar al usuario la existencia del malware al que acompaña u ocultar los puertos abiertos que delaten la comunicación de un atacante con el sistema a través de una puerta trasera [70].

3.8.2. Keylogger

La funcionalidad keylogger permite capturar las teclas pulsadas en el teclado físico del sistema. Puede ser implementado en software o hardware.

4. VISIÓN HISTÓRICA

Desde que apareció el primer malware, este campo ha evolucionado y crecido rápidamente dando lugar a potentes troyanos, virus y otros tipos de malware ya descritos. A continuación se describen algunos de los ejemplos, relacionados con la Criptografía, más notables en la historia del malware:

- **1986 - Cascade Virus**: el primer malware que utilizó cifrado para codificar su contenido fue Cascade. Este código contenía, antes del código principal, una rutina de cifrado simple utilizando la función XOR. Analizando el código en ensamblador del virus se observa que el cifrado aplicado es la función XOR entre el programa principal y la posición de memoria en la que empieza dicho programa [25].

- **1987 - Jerusalem**: el primer virus diseñado para borrar cada programa instalado en los ordenadores infectados. El virus Moctezuma es una de sus variantes cifrada con un algoritmo propio.

- **1988 - Morris Worm**: este gusano infectó a cerca del 10% de los 60.000 ordenadores conectados a ARPANET, incluyendo servidores de la NASA. Morris intentaba averiguar las contraseñas de otros ordenadores para propagarse a ellos. En cuanto a Criptografía, el gusano Morris utilizaba un ataque de diccionario con un fichero cifrado con un algoritmo propio, para que alguien que intentase descompilar el programa no pudiese acceder a él. Comparaba las posibles contraseñas con las almacenadas en el fichero /etc/passwd.

- **1990 - Chameleon**: es el primer virus polimórfico, es decir, cada copia del virus es distinta a las demás, ya que el código muta manteniendo el algoritmo original intacto. Este virus contiene instrucciones que no realizan ninguna función, mezcladas junto a las instrucciones útiles para despistar al criptoanalista. El código de configuración está, en primer lugar, cifrado con una clave aleatoria distinta cada vez que el virus se propaga y, a continuación, partes de ese código cifrado están a su vez cifradas con dos funciones XOR. Finalmente, la última capa de cifrado consiste en repetir el bucle habiendo incrementado los registros previamente. Este virus está basado en el virus Vienna pero con características de cifrado no vistas anteriormente en otros códigos malware.

- **1994 – One-half virus**: este predecesor del ransomware cifra lentamente el disco duro infectado. En cada arranque del disco cifra los dos últimos cilindros no cifrados del disco duro. La clave de cifrado está oculta en el registro de arranque del disco (MBR) que descifrará los datos cifrados si el usuario accede a ellos, evitando ser detectado. En caso de que el virus sea eliminado, la clave de cifrado/descifrado se perderá y no podrá ser recuperada la información.

- **2005 - PGPCoder**: el troyano PGPCoder cifra ficheros con determinadas extensiones (txt, zip, jpg…). Una vez cifrados solicita dinero al usuario si este quiere recuperarlos. El

troyano GPcode, del que deriva PGPCoder, utilizaba un algoritmo de cifrado muy simple que consistía en utilizar la función ADD con una clave de cifrado predeterminada (58 o 0x3a en hexadecimal). Sin embargo, el PGPCoder implementó con éxito un cifrado RSA de 1024 bits que dificultaba la recuperación de los archivos a los usuarios que no quisiesen pagar la *recompensa* [31].

- **2006 – Zeus**: este troyano es uno de los troyanos más importantes a lo largo de la historia del malware. Su código, complejo y sofisticado, ha dado lugar a muchas variantes y a nuevas familias de troyanos a partir del *leak* o filtración de su código en 2011. En la Figura 5 se puede observar que ha dado lugar a familias de troyanos como Murofet y Citadel.

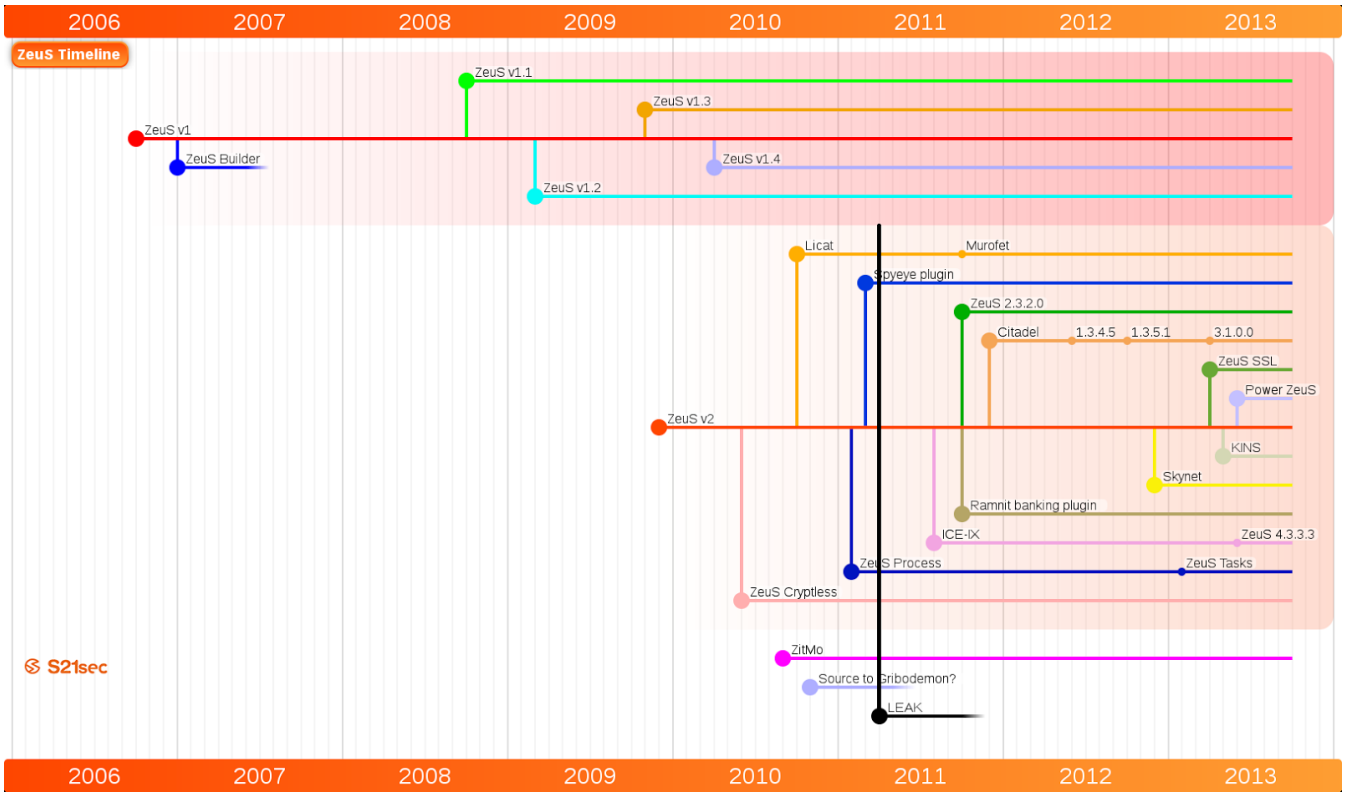

Figura 5. Cronograma de variantes de Zeus

Aunque hay muchas variantes del troyano Zeus, el Zeus original (versiones 1.x y 2.x) se utiliza RC4 tanto para la comunicación con el panel de control como para el cifrado de los ficheros de configuración. Algunas variantes utilizan un cifrado más fuerte, AES. Tanto en el original como en las variantes la clave utilizada es compartida previamente y la define el constructor [55].

Un dato interesante sobre Zeus es la utilización de un algoritmo de compresión poco conocido usado en ciertas secciones de la configuración. Este algoritmo de compresión es NRV2B. Su código se puede encontrar, por ejemplo, en el proyecto Coreboot en Github [26]. Este proyecto se desarrolla con el ánimo de implementar una BIOS *open source* o libre.

- **2007 – Gozi**: este troyano forma una botnet con las máquinas infectadas. La comunicación de los bots con el panel de control está cifrada con un algoritmo de cifrado por bloques personalizado desarrollado por los creadores de Gozi [16].

- **2009 – Mariposa**: la botnet de Mariposa utiliza una serie de esquemas de cifrado. Para la conexión con el panel de control se utilizan paquetes UDP con un algoritmo de cifrado personalizado. Este algoritmo consiste en el uso de la función XOR en cada byte de información alternando dos claves distintas. Para ofuscar las comunicaciones se utiliza un set de comandos en ASCII. Algunos de los comandos observados son *alinfiernoya* y *pillaestenuevoya* <update URL>, utilizados para borrar el bot y actualizar el malware respectivamente [14].

- **2010 – Carberp Trojan**: este troyano, como Zeus, también sufrió una filtración de código pero esto no ocasionó variantes como en el caso anterior a pesar de que el código es también sofisticado y es un malware muy potente [24]. En el caso de Carberp se utilizan dos capas de cifrado. La primera intenta, mediante el uso de distintas claves XOR y rotaciones de bits, evitar el análisis automático del código. Estas claves XOR se generan en función de una clave fija y el índice de memoria de los bytes que se desean cifrar. La segunda capa está basada en instrucciones matemáticas ofuscadas con código basura dificultando más el análisis [52].

- **2010 - Stuxnet**: el gusano Stuxnet es malware poco convencional por su gran tamaño (medio *megabyte*) y por estar escrito en distintos lenguajes de programación como C y C++, por ejemplo. Dentro del ámbito criptográfico, Stuxnet contiene ficheros cifrados con la clave XOR 255 (0xFF). Además, utiliza *Esteganografía* para ocultar una clave de 256 bits que cifra archivos con el algoritmo Rijndael.

5. TIPOS DE CRIPTOGRAFÍA EN MALWARE EN LA ACTUALIDAD

En la actualidad el malware utiliza, por lo general, criptosistemas robustos con los que ocultar, por ejemplo, las comunicaciones entre la máquina infectada y el atacante. La evolución de las técnicas criptográficas en el ámbito del malware queda patente en troyanos como Flame o CryptoWall. Una vez que la mayoría del malware deja de ser utilizado en deterioro de los sistemas infectados y se empieza a utilizar con una finalidad económica, la necesidad de proteger la información valorada económicamente se incrementa. Esta información no consiste únicamente en datos bancarios sino que también se valora información relacionada con hábitos del usuario, por ejemplo. A continuación se analizan los criptosistemas utilizados por distintos malware para proteger su información más valorada.

5.1. Trojan Horse (Troyano)

5.1.1. Ransomware

CryptoLocker

La funcionalidad principal de este tipo de malware es cifrar los archivos del usuario infectado. Por esta razón los algoritmos criptográficos utilizados deberán ser avanzados para asegurar el éxito del malware.

En primer lugar, la conexión entre la máquina infectada y el atacante se realiza mediante HTTP, sin SSL, cifrando los datos con una clave RSA. La clave pública está incrustada en el malware. Una vez realizada la conexión, se solicita otra clave pública RSA de 2048 bit. El malware localiza a continuación ficheros con determinadas extensiones (odt, xls, rtf, etc) y genera una clave AES de 256 bit por cada uno de ellos. Esta clave es utilizada para cifrar el fichero correspondiente. A continuación se cifran todas las claves AES con la clave RSA mencionada previamente [19].

Existe también una variante de CryptoLocker que, en vez de utilizar claves de cifrado RSA de 2048 bit, utiliza el cifrado simétrico TEA (*Tiny Encryption Algorithm*), de tal forma que es posible descifrar fácilmente los archivos utilizando la misma clave utilizada para cifrarlos y eliminando la extensión .cryptolocker añadida al archivo durante el cifrado [20].

CryptoWall

CryptoWall es un malware basado en la familia de CryptoLocker y, como este, genera un par de claves RSA de 2048 bit en el servidor e incrusta la clave pública en el código malicioso. El proceso de cifrado es similar al utilizado en CryptoLocker, también utilizando claves AES de 256 bits [42].

TorrentLocker

TorrentLocker fue descubierto en 2014 como un nuevo ramsonware que era distinto de CryptoLocker y CryptoWall pero usaba algunos de los componentes de estos. Después de realizar una conexión segura con el C&C (*Command and Control* o panel de control) e intercambiar certificados, se cifran archivos con determinadas extensiones. En las primeras versiones de TorrentLocker se utilizaba un sistema de cifrado que permitía el descifrado de los archivos dado que cifraban todos los archivos con la misma clave XOR. Esta clave era generada utilizando funciones de AES (Rijndael) y funciones hash [56]. En versiones posteriores se empezó a utilizar un sistema de cifrado distinto. En estas versiones se utiliza una clave distinta por cada fichero además de cifrar de un modo distinto, mediante un tipo de cifrado en bloques, *Cipher-Block Chaining* (CBC). Otras versiones recientes utilizan cifrado AES de 128, 192 y 256 bits [29]. Estos métodos mejoran sustancialmente el criptosistema utilizado en TorrenLocker.

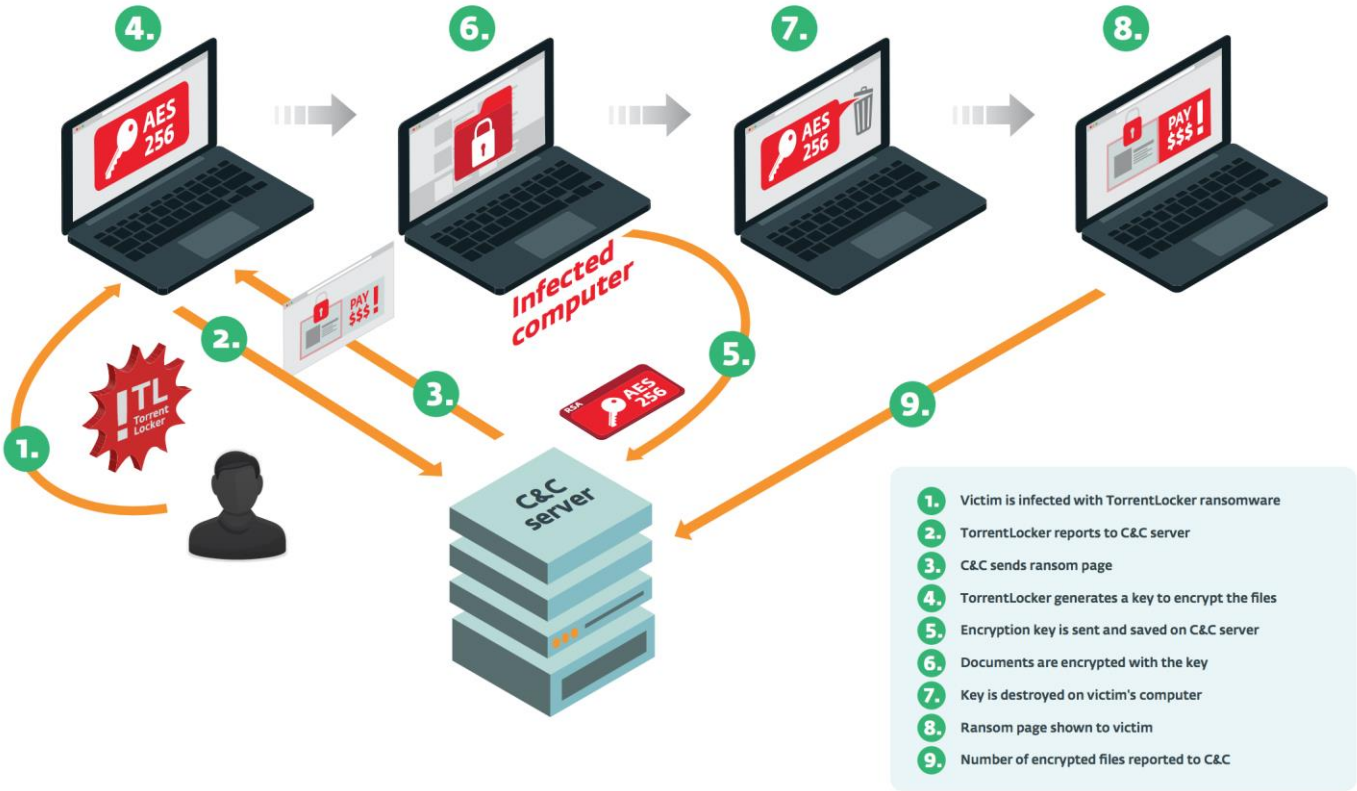

Figura 6. Funcionamiento de TorrentLocker

TorLocker

El código de TorLocker está cifrado con una clave AES de 256 bits que se descifra al ejecutar el troyano. El método de cifrado de archivos utilizado por TorLocker es una clave AES de 256 bits generada individualmente para cada archivo. Después de cifrar el archivo, esta clave es cifrada a su vez con una clave RSA de 2048 bit y añadida al archivo cifrado. Únicamente se cifran los primeros 512MB de cada archivo, en caso de que su tamaño exceda

esa cantidad. Una vez pagado el rescate o *ramson* el malware contacta con el C&C mediante la red Tor para recibir la clave privada RSA de 2048 bits con la que descifrar los archivos. Pese al fuerte criptosistema utilizado por TorLocker, Kaspersky detectó una vulnerabilidad que permite descifrar la mayoría de los archivos cifrados (~70%). Esta empresa ha desarrollado una herramienta que permite descifrar esos archivos sin pagar el rescate: *ScraperDecryptor* [57].

CTB-Locker

CTB-Locker, también conocido como Trojan-Ransom.Win32.Onion, es un troyano que utiliza un criptosistema más seguro que los criptosistemas utilizados por otros ramsonware, que consisten normalmente en la combinación AES + RSA. En este caso el protocolo de Criptografía pública utilizado es ECDH o *Elliptic curve Diffie–Hellman* (ver punto *6.3 Criptografía asimétrica*).

La implementación del criptosistema se compone de los siguientes pasos:

1. En primer lugar se genera un par de claves pública-privada (*master-public* y *master-private*) en la máquina infectada. La clave privada se transmite de forma segura al servidor y no se almacena en la máquina.
2. A continuación, se genera otro par de claves pública-privada (*session-public* y *session-private*) por cada fichero.
3. Sabiendo la clave privada *session-private* y la clave pública del C&C *master-public* puede calcularse un punto P de la curva que podrá ser también calculado en el C&C, siempre que el panel de control posea la *master-private* y la *session-public*. Este punto P es entonces un secreto compartido por la máquina y el panel de control.
4. A continuación se utiliza el secreto compartido para generar una clave de 256 bits. Para ello se aplica al secreto compartido la función hash SHA-2 de 256 bits.
5. El cifrado de cada archivo se realiza con el algoritmo AES utilizando la clave de 256 bits calculada. Previamente se comprime el fichero con el algoritmo Zlib.
6. Después del cifrado, la clave pública de cada fichero *session-public* se guarda en él y la clave privada *session-private* se descarta. El secreto compartido tampoco se guarda en la máquina infectada.

El protocolo ECDH cumple la siguiente igualdad en la que el secreto compartido es el punto P calculado sobre una curva elíptica:

Secreto compartido = ECDH (*master-public*, *session-private*) = ECDH (*master-private*, *session-public*) [58]

Es decir, el punto P calculado con *master-public* y *session-private*, será el mismo punto P calculado con *master-private* y *session-public*. La clave *sesión-private* del archivo ha sido eliminada. Por esta razón sólo se puede calcular de nuevo el punto P en el C&C, que posee la *master-private* y la *session-public*. Cuando el panel de control necesite calcular la clave que

ha cifrado un archivo calculará el secreto compartido o punto P y generará el su hash de 256 bits utilizando SHA-2.

Por otra parte, en caso de interceptar la comunicación con el C&C en la que se manda la clave *master-public* no será tampoco posible obtenerla para descifrar los archivos. Esto se debe a que la comunicación con el panel de control está también cifrada siguiendo el mismo criptosistema ECDH utilizado para el cifrado de archivos.

En la comunicación con el C&C los pasos seguidos son similares a los anteriores:

1. El C&C genera un par de claves *server-private* y *server-public* e incrusta la clave pública en el malware.
2. Al establecer la conexión desde la máquina ya infectada, el propio malware genera un nuevo par de claves *client-public* y *client-private*. Además calcula el secreto compartido, es decir, el punto P, de la forma descrita previamente con la clave *client-private* y la clave *server-public*.
3. A continuación, el malware cifra los datos que desea transmitir al C&C usando como clave el hash SHA-2 de 256 bits del secreto compartido y cifrado AES de 256 bits.
4. La clave *client-public* es enviada sin protección al panel de control antes de borrar de la máquina infectada el par de claves *client-public*, *client-private* y el secreto compartido.

El descifrado de los datos transmitidos lo puede realizar únicamente el panel de control que posee *server-private* y *client-public*. Estas claves conocidas permiten calcular el punto P y, a partir de él, la clave generada con el algoritmo SHA-2 de 256 bits. Gracias al empleo de ECDH, el CTB-Locker es conocido por su seguridad criptográfica [58].

Otros

Otros ramsonware conocidos cifran los archivos con claves de 256 bits con cifrado AES. Algunos de ellos son *Bit Cryptor* y *Los Pollos Hermanos crypto virus*. El ramsonware *ABOUT FILES!* utiliza, además de AES de 256 bits, claves RSA de 2048 bits de forma parecida a los explicados anteriormente con detalle [15]. Este mismo método es usado también por el ramsonware *Alpha Crypt*, evolución de *Tesla Crypt* [7].

5.1.2. Downloader

Bublik

En las primeras versiones detectadas de este downloader la conexión con el C&C se hacía mediante HTTPS pero, a partir de 2014, todas las versiones pasaron a descargar un archivo comprimido con Lempel-Ziv (LZ) y cifrado con una clave XOR fija incrustada en el código del troyano [22]. Una constante incrustada en código, por ejemplo esta clave XOR, es denominada constante *hardcodeada*.

5.1.3. Click-fraud

MIUREF

Este troyano cifra las conexiones con el panel de control utilizando un cifrado RSA codificado en base64 [67].

5.1.4. DDOS

Beta Bot

Beta Bot utiliza múltiples capas de cifrado teniendo cifrado incluso el código utilizado para descifrar los datos de configuración. Este código está cifrado con una clave hardcodeada RC4. Una vez descifrado, se puede proceder a descifrar los datos de configuración cifrados también con RC4 pero con una clave distinta que es generada por los bytes pares de un bloque de 32 bytes de un archivo determinado [4].

5.1.5. Backdoor

Poison Ivy

Este troyano implementa un protocolo de conexión complejo. La mayor parte de la comunicación está cifrada utilizando el cifrado Camellia de 256 bit. La clave se deriva de una contraseña introducida en el panel de control ("admin" por defecto) en codificación hex-ASCII y rellenando con ceros (*zero-padding*) hasta 256 bits [21].

ZeroAccess

Este backdoor cifra las conexiones con el panel de control con un cifrado RC4 con una clave incrustada en su código. La clave RC4 de la respuesta (las peticiones de archivos no son cifradas) es el hash MD5 de los 12 bytes que componen la petición. En la segunda versión de este troyano se combina este cifrado RC4 con la función XOR usando también la clave *ftp2* hardcodeada. La conexión se realiza por UDP [35].

5.1.6. Memory Scrapper

GetMyPass

GetMyPass es un troyano que afecta a los *terminales puntos de venta* o TPV (PoS o *Points of Sale* en inglés). Este troyano, como BackOff, utiliza cifrado RC4 para guardar la información obtenida de las tarjetas de crédito utilizadas en el TPV. Para la validación de los datos obtenidos utiliza el algoritmo matemático de Luhn [79] que es utilizado también en otros malware PoS como *FrameworkPoS* y *Dexter* [13].

Backoff

Este malware está diseñado para robar credenciales de acceso bancarias en los PoS. El malware se hace pasar por una carpeta falsa de Oracle Java. El algoritmo de cifrado personalizado utilizado por este malware usa una clave hardcodeada. La función XOR se aplica a los datos después de desplazarlos un determinado número de posiciones alternativamente a izquierda y derecha [39].

5.1.7. Troyano bancario

Boleto

La configuración de este troyano va cifrada con una clave XOR de 32 bits hardcodeada en el propio binario del troyano. Además de cifrarse, la configuración va comprimida con el algoritmo de compresión ZLIB [60].

Citadel

Citadel, basado en el troyano Zeus, combina los criptosistemas utilizados por este en sus diferentes versiones, RC4 y AES. El criptosistema resultante es más potente que la utilización de los criptosistemas anteriores por separado y permite ofuscar el proceso de ingeniería inversa.

La utilización de los algoritmos de cifrado varía de una versión de Citadel a otra. Por ejemplo, en la versión 1.3.4.5 se genera una clave AES a partir de una clave RC4. Esta clave AES permite generar otra clave RC4 de comunicación. Sin embargo, en la versión 1.3.5.1, la clave AES se genera con un algoritmo AES modificado usando la función XOR.

Otro ejemplo significativo es la versión 3.1.0.0 que utiliza un algoritmo AES modificando usando, además de la función XOR, la función SALT_KEY. Esta última función genera datos aleatorios añadidos al final del resultado del algoritmo para dificultar los ataques por fuerza bruta. En todas las versiones el vector de inicialización se modifica utilizando esta función SALT_KEY [55].

Cridex

Cridex, llamado también Feodo o Bugat, ha ido mejorando los criptosistemas utilizados a lo largo de las versiones detectadas los últimos años. En las primeras versiones del troyano los datos de configuración solicitados al C&C no iban cifradas. Otra medida de seguridad para evitar la conexión al C&C por parte de una máquina no infectada es la utilización de un *blob* (*binary large object*) utilizado para autenticarse.

Las versiones posteriores pero previas a 2014 se comunican con el panel de control cifrando los datos con una clave RC4 que se genera de forma aleatoria y que se envía como parte de la comunicación cifrada con una clave pública RSA que va hardcodeada dentro del

troyano. Los datos de configuración devueltos por el panel de control se descifran con la misma clave RC4 con la que se cifra el *payload*, o carga útil de las peticiones.

A diferencia de las anteriores, las versiones detectadas en 2014 se comunicaban con el panel de control generando un payload que se empaquetaba con gzip y cuyos primeros 8 bytes eran sustituidos con bytes aleatorios para evitar que fuese fácil descifrar la información enviada. Los datos de configuración se descifran restaurando la cabecera del gzip (0x1f, 0x8b, 0x08, 0x00, 0x00, 0x00, 0x00,0x00) que también viene sustituida del servidor [16].

Dyre

Las peticiones de la máquina infectada al panel de control para solicitar órdenes o nuevas configuraciones no están cifradas pero las URL utilizadas incluyen información relativa a la botnet y a la máquina para evitar que alguien pueda acceder a los C&C de la botnet por error. Sin embargo, las respuestas del C&C utilizan cifrado AES cuya clave es generada haciendo múltiples iteraciones con la función SHA-2 de 256 bits sobre los datos de la respuesta. La clave AES está incrustada en esa misma respuesta, por lo que puede ser descifrada fácilmente. Este troyano firma digitalmente los archivos de configuración y los plugins malware para evitar su manipulación. El tráfico HTTP va cifrado con diferentes algoritmos de cifrado dependiendo del tipo de paquete transmitido [8] [17]. Al conectarse la máquina infectada a la botnet, registra una clave pública RSA en esa botnet y solicita los ficheros de configuración.

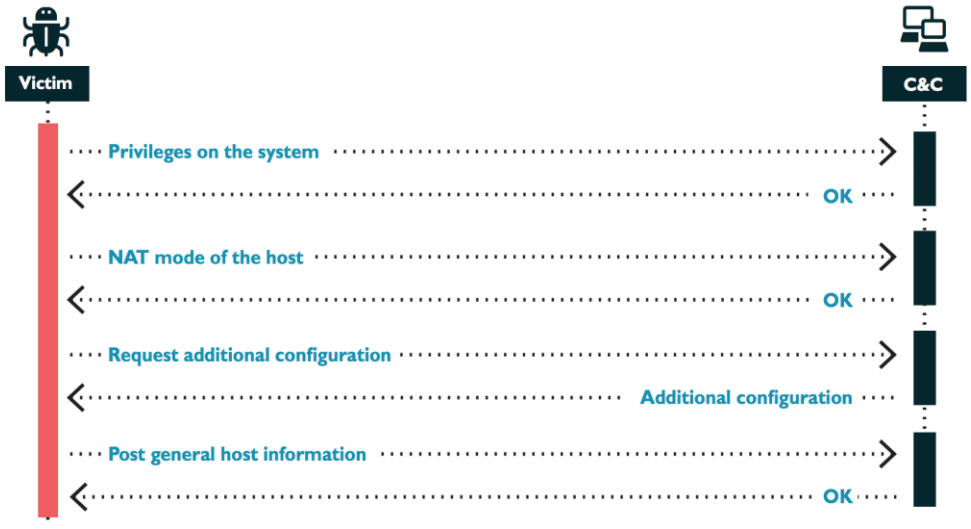

Figura 7. Comunicación entre máquina infectada con Dyre y el C&C

Spyeye

Este conocido troyano utiliza un criptosistema débil en la comunicación con el panel de control, al contrario que otros troyanos bancarios de índole parecida. El criptosistema consiste en la compresión Zip de archivos y el uso del cifrado Zip estándar [46].

Hermes

Hermes es un troyano que comparte muchas características con Spyeye pero entre estas características compartidas no se encuentra el criptosistema utilizado. Al contrario que Spyeye, Hermes ofusca las conexiones con el C&C mediante la función XOR. Además, cifra los archivos guardados en disco mediante el algoritmo DES [46].

Hesper

La configuración de este troyano se cifra con el algoritmo Twofish. La clave utilizada se genera a partir de un hash HMAC-SHA-2 de 512 bits, que es un método de autenticación que utiliza una clave en conjunción con una función hash. La función hash se aplica sobre los ficheros de configuración generando así la clave de cifrado. La información de la máquina infectada: nombre de usuario, clave de cifrado, módulos maliciosos descargados, etc. se encuentran guardados en un fichero dat [40].

Kins

Al igual que ocurre con Citadel los algoritmos de cifrado utilizados han ido cambiando según ha ido evolucionando el troyano. Dependiendo de la versión, la comunicación con el panel de control se efectúa con RC4 o con una combinación de AES y Visual Encrypt. Este último algoritmo se utiliza como medio de ofuscación. Su implementación se puede observar en la siguiente referencia [83]. El cifrado de los ficheros de configuración se realiza, según la versión del troyano con: Base64 + RC4, Base64 + RC6 o AES + XOR. Existe también una versión de Kins que emplea *esteganografía*, es decir, que oculta mensajes dentro de otros objetos, por ejemplo incluir algunos bytes incrustados en una imagen. En este caso, los códigos de configuración están ocultos en una imagen JPG [38].

Kronos

Este troyano cifra las comunicaciones con el panel de control con una combinación de la función XOR y el algoritmo AES. Además, los ficheros de configuración se cifran con AES.

Shylock

Este troyano cifra sus comunicaciones con el panel de control con una combinación de RC4 y Base64. Los ficheros de configuración van cifrados con el mismo criptosistema [16].

Tinba

Las comunicaciones entre la máquina infectada con este troyano y el C&C están cifradas con una clave XOR incrustada y, posteriormente, con el algoritmo RC4 cuya clave está también incrustada. La configuración va cifrada también con RC4 [5].

GameOver Zeus

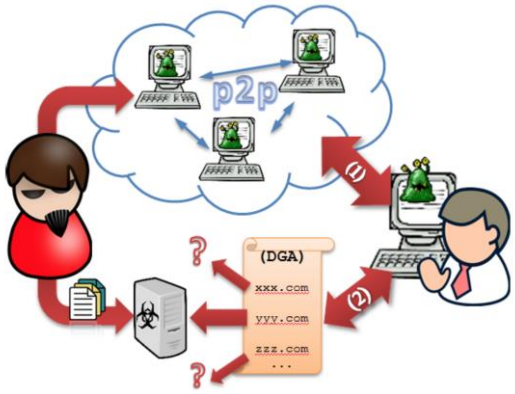

Figura 8. Funcionamiento de Gameover Zeus

Este troyano, también llamado Murofet, está basado en Zeus y basa su estructura de red en una red P2P. Las redes P2P no se han utilizado en ninguna botnet previa conocida. Una de las características propias de las redes P2P es el intercambio directo de recursos entre los nodos sin depender de un control intermedio y esto supone un riesgo de seguridad si no se toman las medidas oportunas. Por este motivo Gameover Zeus utiliza un sistema de firma para el intercambio de datos entre nodos. La única clave privada que puede firmar la información la posee el *botmaster*. La clave pública, sin embargo, está presente en todos los *bots* pero cifrada con un cifrado simple basado en la función XOR. Por otra parte, los paquetes TCP de las conexiones son además cifrados con RC4. Cada sesión TCP utiliza dos claves: una para el remitente y otra para el receptor. Ambas son generadas en función del ID del nodo del destinatario [16]. Al contrario que en otras variantes de Zeus, Murofet no cifra el contenido de las peticiones POST al panel de control [9].

Además, Murofet utiliza un sistema de cifrado similar al del troyano Boleto, analizado más adelante, para cifrar archivos, es decir cifrado con una clave predefinida XOR de 32 bit. La clave XOR se genera en función de la ID de sección (SID), el tamaño de la sección y la versión del fichero de configuración. Antes del cifrado el fichero es comprimido como en el caso de Boleto [72].

Neverquest

Neverquest o Vawtrack es un troyano evolucionado a partir de Gozi, descrito anteriormente. Utiliza un criptosistema personalizado como en su predecesor. El criptosistema en cuestión está basado en XTEA, una evolución del algoritmo TEA que corrige algunas de sus debilidades, utilizando una clave hardcodeada de 128 bit. Este algoritmo es utilizado en el cifrado de la configuración. Además, para proteger ciertas cadenas de caracteres y las comunicaciones con el C&C se utiliza un cifrado que intenta actuar como el cifrado de Vernam (que teóricamente no es posible romper). Neverquest utiliza una simplificación de este criptosistema usando un LCG (*Linear Congruential Generator*). El LCG genera cadenas de números pseudoaleatorios basados en el valor de la semilla, como se puede observar en el código de la Figura 9. Esta semilla es generada a partir de datos propios de la máquina infectada, por ejemplo a partir de la dirección MAC de la máquina. Después es enviada junto al fichero cifrado [6].

```
unsigned int random(unsigned int *seed) {
    *seed = 0x343FD * *seed + 0x269EC3;
    return (*seed >> 16) & 0x7FFF;
}
```

Figura 9. Código de cifrado de Neverquest

Otros

Threat Finder es un troyano que cifra los ficheros por los que pide recompensa con una clave RSA de 2048 bit generada específicamente para cada máquina. Al contrario que CryptoWall, este troyano no elimina las copias ocultas generadas por Windows (*Shadow Volume Copies*), por lo que existe la posibilidad de recuperar entera o parcialmente los archivos cifrados [15]. Otro ramsonware que utiliza el mismo criptosistema es *Key holder*. Otra botnet conocida es *Ice IX* o *Ice9* que utiliza un cifrado RC4 para ocultar sus ficheros de configuración [16].

5.2. PUP

Adware.Mac.InstallCore.1

Este adware es distribuido en forma de ficheros javascript. Estos scripts son, en algunos casos, cifrados con el algoritmo AES. Al establecer conexión con el panel de control se solicita, además de los ficheros de configuración, un fichero cifrado con la función XOR y comprimida usando GZIP [18].

Careto

Este malware, según Kaspersky, está dirigido a instituciones de gobiernos, compañías petrolíferas, embajadas, etc. Una de las razones principales para hacer esta afirmación es que Careto emplea técnicas de ofuscación y ocultación nunca vistas en malware con objetivos diferentes. Algunas fuentes afirman que este malware es de procedencia española, debido a su nombre y al uso de algunas cadenas de caracteres en castellano, incluida una clave de cifrado RC4: *Caguen1aMar*.

Figura 10. Careto

Para descifrar el payload de las peticiones al panel de control se utiliza una clave fija RC4. Este payload está compuesto por tres bloques, cada uno cifrado por separado pero con la misma clave. Careto utiliza diferentes vulnerabilidades en distintos sistemas operativos. Por ejemplo, contiene un módulo que hace la función de backdoor para el sistema operativo OS X, de Apple. En este módulo se utiliza un cifrado simple XOR para ocultar todas las cadenas importantes de caracteres. La comunicación de este módulo backdoor con el C&C en este caso se cifra con AES usando SHA-1 para la autenticación cruzada. El SHA-1 se calcula a partir de la ID de usuario concatenada dos veces.

Sin embargo, el criptosistema utilizado por el propio Careto para las comunicaciones con el panel de control es diferente, ya que se basa en dos capas de cifrado. La información recibida del panel de control se cifra con una clave AES temporal que también es cifrada pero con cifrado RSA. La misma clave RSA sirve para cifrar los datos de respuesta al C&C [34].

5.3. Virus

Ramnit

Ramnit es un virus con componente bancaria que utiliza técnicas de *bitshifting* y manejo de excepciones para impedir el análisis estático del binario. Además, utiliza un cifrado basado en la función XOR y crea un *handler* o gestor para el manejo de excepciones que le permiten desempaquetarse. Por otro lado, el binario está empaquetado con UPX debajo de las capas de cifrado anteriores [45].

5.4. Worm

Flame

Flame, también conocido como Skywiper, es un malware complejo compuesto de muchas herramientas. En su código se utilizan al menos 5 métodos diferentes de cifrado y 3 técnicas de compresión distintas. Alguno de estos métodos de cifrado no se ha podido relacionar con métodos de cifrado conocidos, como el E1 descrito en la referencia [36]. Este método de cifrado consiste únicamente en el uso de una tabla de sustitución de cifras de dos o tres dígitos por otras cifras. El siguiente método de cifrado de flujo, nombrado E2 en el informe, consiste en aplicar un algoritmo matemático (sumas, multiplicaciones y potencias) con desplazamientos de bits.

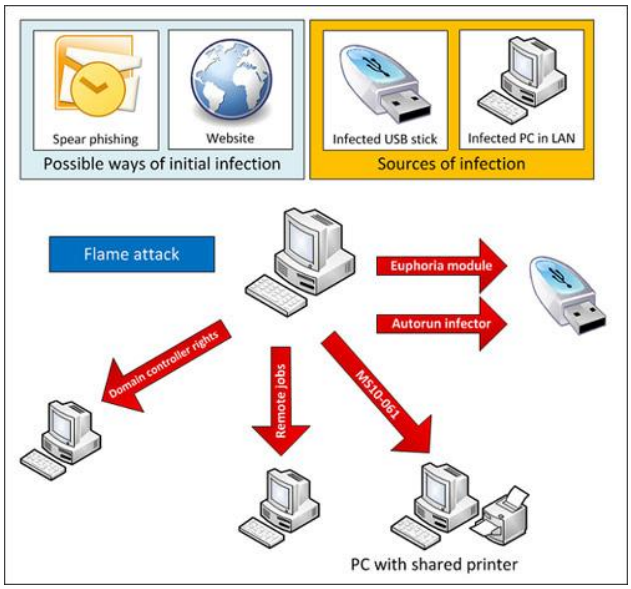

Figura 11. Ataque de Flame

El método E3 es otro cifrado de flujo que aplica otra serie de funciones matemáticas combinadas. El método E4, sin embargo, utiliza funciones aplicadas a los punteros a memoria, aunque no se ha encontrado evidencias de que se use este método en ninguno de los archivos que componen el malware. Por otra parte, el método E5 consiste en aplicar la función XOR con una determinada clave predefinida. La función XOR es utilizada para cifrar varios archivos con diferentes claves. Finalmente, el método E6 combina la función XOR con una tabla de sustituciones monoalfabéticas, reconstruida parcialmente en el informe. Este último método

utiliza como parámetros el tamaño o longitud de los datos cifrados y la dirección de memoria en la que empiezan.

En este otro informe [3] se describe otro algoritmo de cifrado basado en DES utilizando el cálculo circular de 16 expresiones que caracteriza a este conocido algoritmo. Los archivos del cliente recibidos en el servidor son almacenados en una base de datos MySQL cifrados usando el algoritmo Blowfish en su modo CBC con un vector inicial estático. La clave Blowfish es generada aleatoriamente para cada archivo y cifrada con un criptosistema asimétrico proporcionado por la función de PHP *openssl_public_encrypt* [59].

Qakbot

Los datos robados por este gusano se transmiten al C&C cifrados mediante una combinación de la función XOR y una rotación a la derecha. Los ficheros de configuración también están cifrados [68].

Scanned Document Attached Spam

Este gusano se propaga a través de correo basura ofreciendo un fichero zip cifrado al destinatario del correo. Se ofrece una contraseña para descifrar el archivo que contiene un ejecutable malicioso. El malware se conecta entonces con un servidor localizado en Reino Unido transmitiendo ciertos datos de la máquina infectada. El cifrado utilizado es personalizado, probablemente utilizando la función XOR. Otros malware similares son *Fake "Outlook Settings" Spam* o *Citigroup Secure Message Spam* [65].

Asprox

Asprox, también conocida como Kuluoz, es una botnet que en sus inicios fue utilizada como *password stealer* pero que fue actualizada para que emplease ficheros maliciosos en el spam enviado para expandirse. El criptosistema integrado para la conexión con el C&C es cifrado RC4 cuya clave es generada a partir del SID del ordenador infectado. Los 8 primeros caracteres del hash MD5 de ese SID compondrán la clave de cifrado. A continuación se utiliza además cifrado RSA en el fichero que contienen las claves RC4. En versiones posteriores se ha actualizado el criptosistema. La diferencia más notable es la compresión de la petición POST con el algoritmo de compresión *BZ2* y su posterior cifrado con una clave aleatoria de 16 bytes. Por tanto, en estas versiones, la clave ya no es generada a partir del SID de la máquina [66].

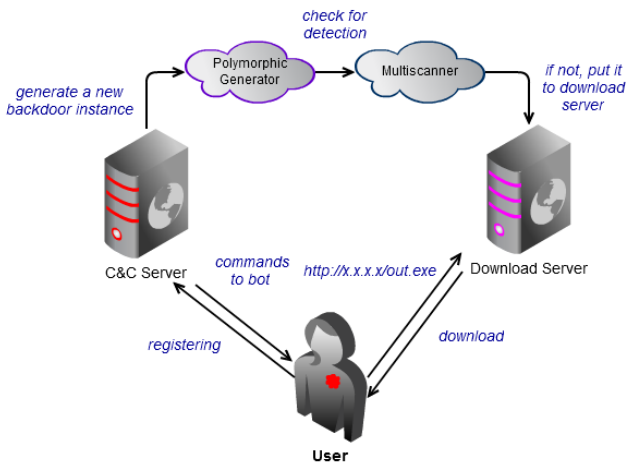

Figura 12. Ataque de Shiz

5.5. Hacking Tools

5.5.1. RAT

Shiz

Este software malicioso evita ser detectado utilizando un *generador polimórfico*, es decir, que cada vez que se envía el malware desde el C&C a una máquina se vuelve a empaquetar y a cifrar con una clave diferente. De esta forma se consigue que el hash correspondiente al fichero cambie de una máquina infectada a otra y se dificulta su detección [37].

Fakem RAT

Esta herramienta maliciosa evita la detección de su conexión entre la máquina infectada y el panel de control mezclando su tráfico con tráfico legítimo. Además, esta conexión se cifra con un algoritmo personalizado. Cada byte se cifra con la función XOR con cada una de las letras de un string (*YHCRA*). La información, es transmitida en blobs de 1024 bytes camuflados con cabeceras de tráfico legítimas [74].

5.6. Otros

Regin

Este complejo software está formado por diferentes etapas, representadas en la Figura 13, algunas de las cuales no son necesarias, por lo que es personalizable. En concreto, la etapa 3 contiene las rutinas de cifrado y descifrado entre otras herramientas. El *sistema virtual de archivos cifrados* (*Encrypted virtual file system* o EVFS) contiene archivos cifrados con una variante del algoritmo de cifrado RC5 que utiliza bloques de 64 bits y 20 rondas [69].

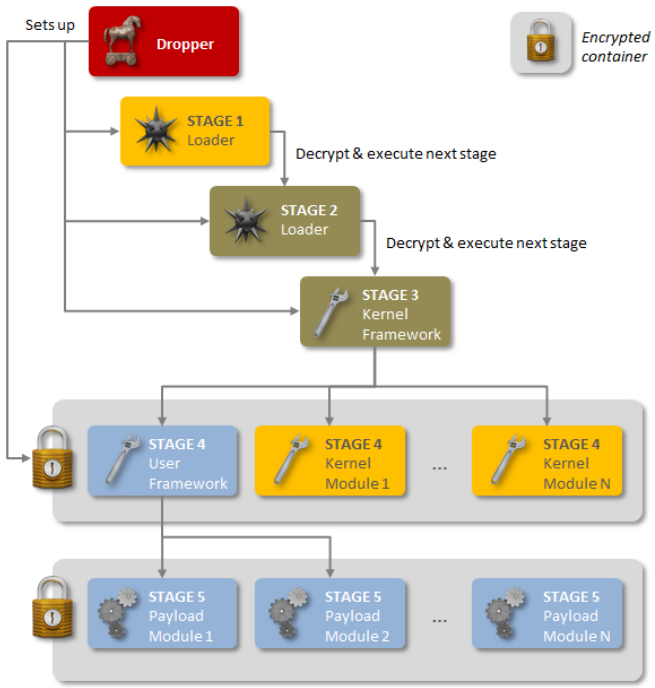

Figura 13. Proceso de infección de Regin

6. SEGURIDAD CRIPTOGRÁFICA EN LOS ESQUEMAS ACTUALES

La seguridad de la información es uno de los activos más importantes de compañías y gobiernos en la actualidad. Existe mucha información privada en formato digital y para garantizar su seguridad se recurre a la Criptografía. La Criptografía tiene, como se ha indicado en la introducción, tres posibles finalidades: confidencialidad, autenticidad e integridad. En ocasiones no es posible garantizar absolutamente estas finalidades. Por esta razón se definen varios tipos de seguridad criptográfica.

6.1. Tipos de seguridad criptográfica

- *Seguridad incondicional*: el atacante no puede averiguar la información cifrada aunque tenga tiempo y recursos computacionales ilimitados. Por ejemplo, se considera que el cifrado de Vernam es incondicionalmente seguro, ya que no se conoce en la actualidad ninguna forma de romperlo.
- *Seguridad computacional*: el sistema es seguro frente a un atacante con tiempo y recursos computacionales limitados. Por ejemplo, los problemas del logaritmo complejo en cifrado asimétrico proporcionan una seguridad computacional razonable, dependiendo de la longitud de la clave empleada.
- *Seguridad probable*: un criptosistema cuya integridad no se puede demostrar, pero que no ha sido violado.
- *Seguridad condicional*: todos los demás sistemas están englobados en esta categoría. Son seguros siempre que el enemigo carezca de medios para atacarlos [23].

6.2. Criptografía simétrica

El uso de Criptografía simétrica es seguro en sistemas en los que el intercambio de claves sea seguro o en los que una entidad conoce y maneja todas las claves [23]. En el ámbito del malware, es posible usar Criptografía de clave secreta cuando es necesario cifrar archivos de forma local con una clave proporcionada por el C&C, que es la entidad que conoce y maneja las contraseñas [80].

Ventajas generales

- Eficaz tanto en hardware como en software: alta velocidad de cifrado y descifrado.

Desventajas generales

- Necesidad de un canal seguro para el manejo de claves.
- Necesidad de almacenar un número alto de claves y de protegerlas convenientemente.

- Vulnerable a ataques de diccionario: intentos de averiguar la clave probando palabras pertenecientes al diccionario o a un conjunto de claves probables. Se prueban también combinaciones con letras mayúsculas y minúsculas o se añaden números y símbolos.
- Vulnerable a ataques de fuerza bruta: intentos de averiguar la clave probando todas las combinaciones posibles. Se intenta evitar realizar estos ataques probando primero ataques de diccionario y ataques lineales (análisis o diferenciales que reducen las combinaciones a las más probables).

Algoritmos de clave simétrica

DES (Data Encryption Standard)

Es uno de los sistemas más empleados y extendidos, por lo que es de los más probados. Además, su implementación es sencilla y rápida. Sin embargo, tiene el inconveniente de no permitir una clave de longitud variable; por lo cual no es posible aumentarla para tener más seguridad. También tiene el inconveniente de ser vulnerable a un ataque diferencial, ya que las posibilidades se reducen a 2^{47}. Aunque no se pudiese realizar un ataque diferencial, la clave de 56 bits es demasiado corta. Existe una mejora de este algoritmo, *3DES*, que consiste en aplicar el algoritmo DES tres veces con claves más largas. Sin embargo, este algoritmo no ha sido encontrado en el malware analizado en este trabajo [54].

RC4

Este sistema de cifrado de flujo es uno de los más utilizados en los malware analizados en este trabajo. Las principales ventajas de este algoritmo son su velocidad y simplicidad. Además, su implementación tanto en hardware como en software es sencilla y requiere de pocos recursos para obtener un buen rendimiento. Sin embargo, RC4 fue excluido de los estándares de alta seguridad debido a que algunos modos de uso lo convierten en un sistema muy inseguro. Por ejemplo, en WEP se implementa el vector de inicialización de forma predecible de un paquete a otro, por lo que se puede atacar con facilidad una red Wi-Fi con una clave de este tipo. En caso de utilizar RC4 es importante no usar la misma clave para cifrar dos datos distintos, ya que la clave puede averiguarse fácilmente por distintos métodos [32].

RC5

En este sistema de cifrado por bloques se aplican operaciones XOR a los datos con claves de tamaño variable (32, 64 o 128 bits). Además permite un número variable de iteraciones que mejoran exponencialmente la seguridad criptográfica del algoritmo al aumentarlas. Otra ventaja es su utilidad como generador de números aleatorios. En Regin se utiliza, por ejemplo, una variante de RC5 que utiliza bloques de 64 bits y 20 rondas, que incrementa las 12 rondas sugeridas originalmente para la implementación de este algoritmo [54].

RC6

El algoritmo RC6, empleado en algunas versiones de Kins, está basado en RC5. Al igual que su predecesor, RC6 puede ser parametrizado para soportar una amplia variedad de longitudes de palabra, tamaños de clave y número de rondas. Además de las operaciones utilizadas en RC5, RC6 añade una operación extra de multiplicación haciendo que las rotaciones sean dependientes de cada bit en una palabra y no únicamente los bits menos significativos.

AES (Advanced Encryption Standard)

Este algoritmo, muy utilizado por los ramsonware para cifrar archivos, permite utilizar claves de longitud variable (128, 192 o 256 bits). AES opera con bloques de bytes a los que aplica distintas operaciones matemáticas como sustituciones no lineales de bytes, sumas XOR intermedias y desplazamientos. El número de rondas depende del tamaño de la clave: 10 rondas para claves de 128 bits, 12 rondas para claves de 192 bits y 14 rondas para claves de 256 bits. En 2006 fue posible romper el algoritmo en 7 rondas para claves de 128 bits, en 8 rondas para claves de 192 bits y 9 rondas para claves de 256 bits. Por este motivo algunos expertos en seguridad opinan que no es conveniente confiar en su seguridad [54].

El modo de cifrado por bloques encontrado en los malware descritos previamente es CBC (Cipher-block chaining) en el que el cifrado de un bloque depende del bloque cifrado anterior. Para el primer bloque se emplea un *vector de inicialización*, generalmente aleatorio. La desventaja de este método es su secuencialidad, que no permite que se paralelice [77].

TEA (Tiny Encryption Algorithm)

La implementación de este algoritmo es sencilla y simple. A diferencia de otros algoritmos no permite bloques de tamaño variable ni claves de tamaño variable, utiliza bloques de 64 bits y claves de 128 bits. Es posible implementar un número de rondas variable para mejorar su seguridad aunque el número aconsejado es 64 rondas. La mayor vulnerabilidad de este algoritmo es la existencia de claves equivalentes, ya que cada clave es equivalente a otras tres, lo que reduce la longitud efectiva de su clave a 126 bits. Debido a las vulnerabilidades de TEA, es posible descifrar fácilmente los archivos cifrados por la variante de Crypto-Locker que utiliza este algoritmo [2].

Una de las revisiones de TEA es XTEA, utilizada por Neverquest. Como el anterior, XTEA es un algoritmo de cifrado de bloques, pero en este caso se han corregido las vulnerabilidades con cambios, por ejemplo, en la elección de claves y en la realización de los desplazamientos

Cifrado de Vernam

La seguridad de este cifrado radica en que la clave es de un solo uso (*one-time-pad*). El cifrado de Vernam consiste en utilizar la función XOR para cifrar bit a bit toda la información, necesitando una clave del mismo tamaño que el texto que se desea cifrar. Este cifrado se utiliza en el malware Neverquest pero la implementación, al no ser la clave aleatoria sino pseudoaleatoria, no es segura incondicionalmente. La desventaja principal del cifrado de Vernam es la generación de una clave aleatoria suficientemente larga que es necesario cifrar también. De esta forma la generación y protección de la clave se convierte en un problema cíclico, por lo que no es posible utilizarlo. La variante con LCG utilizada en Neverquest es una posibilidad más realista [23].

Camellia

Este algoritmo de cifrado de bloques tiene la ventaja de que las operaciones de cifrado y descifrado son muy semejantes. Permite utilizar claves de distintas longitudes (128, 192 o 256 bits) y bloques de 128 bits [1].

Blowfish

Blowfish es un cifrado de bloques que permite el uso de una clave de longitud variable, entre 32 y 448 bits. Este cifrado es más rápido que otros como DES e IDEA, además de que su uso no requiere licencia. El gusano Flame lo utiliza para el cifrado de archivos en modo CBC [61].

Twofish

Como Blowfish, Twofish realiza un cifrado de bloques y también permite el uso de una clave de longitud variable, aunque en este caso la clave tiene entre 128 y 256 bits. Una de sus ventajas es, como se demuestra en [62], que es inmune a ataques diferenciales, por lo que no se puede reducir la longitud efectiva de su clave con un ataque de este tipo. Otra de sus ventajas es su optimización para CPU de 32 bit, muy utilizadas en los últimos años. La utilización de este cifrado tampoco requiere de licencia [63].

Cifrado Zip Estándar

El cifrado Zip estándar, utilizado en el malware Spyeye, es conocido por ser relativamente débil, en comparación con el otro cifrado, AES, que permite utilizar, por ejemplo, el programa WinZip [82].

Cifrados personalizados

Los algoritmos de cifrado personalizados propios de algunos malware se basan en el uso de la operación XOR, que puede estar combinado con desplazamientos. El uso de desplazamientos variables y el uso de sal añadida al final de la clave aportan mayor seguridad.

Por el contrario, será menos seguro utilizar una operación XOR con desplazamiento fijo o que desplaza alternativamente a derecha e izquierda, como el usado en Backoff, o incluso una XOR sin ningún desplazamiento.

Otros códigos maliciosos emplean el uso de tablas de sustitución de cadenas de caracteres. Esta práctica es poco segura y debe ser usada únicamente en combinación con otros algoritmos.

6.3. Criptografía asimétrica

Los diferentes criptosistemas de Criptografía asimétrica, que por cuestión de espacio no se describen detalladamente aquí, presentan ciertas ventajas e inconvenientes frente a los de Criptografía simétrica. Se presentan a continuación las ventajas e inconvenientes de los criptosistemas asimétricos utilizados en los malware descritos.

Ventajas generales

- No es necesario establecer un canal secreto seguro para compartir la clave pública.
- El número de contraseñas necesarias es menor que en el caso de la Criptografía simétrica: 2*n frente a n*(n-1)/2, siendo *n* el número de entidades con contraseña.
- Permite evitar el no-repudio gracias a la firma digital.

Desventajas generales

- El tiempo de computación necesario para cifrar con estos criptosistemas es mucho mayor que el necesario en criptosistemas de clave privada. Se estima que los criptosistemas de clave pública necesitan mil veces más tiempo computacional.
- Las claves necesarias son, por lo general, más largas que las de los criptosistemas de clave pública.
- Es conveniente proteger más la clave privada que las claves secretas de los criptosistemas simétricos. Esto se debe a que una clave privada no sólo permite el descifrado de lo que se cifra con su pareja pública, sino que puede ser utilizado como firma digital para suplantar la identidad del dueño de la clave. Por ejemplo, poseer la clave privada con la que firmar el software de actualización de Microsoft puede dar lugar a la infección de millones de dispositivos.

Protocolos y criptosistemas asimétricos

Cambio de claves de Diffie-Hellman

Diffie-Hellman se utiliza para generar una clave privada simétrica en ambos extremos de un canal de comunicación inseguro [23]. La seguridad del intercambio de claves de Diffie-

Hellman radica en la seguridad del logaritmo discreto. Una desventaja de Diffie-Hellman es su vulnerabilidad a un ataque *man-in-the-middle*, ya que este protocolo no proporciona autenticación durante el intercambio de claves [53]. El ataque man-in-the-middle se explica más adelante.

Criptosistema RSA

La seguridad de este algoritmo radica en el problema de factorización de números muy grandes obtenidos a partir de la multiplicación de dos números primos no conocidos por el atacante. En el caso de RSA, las claves necesarias para garantizar la seguridad son largas: por lo general se utilizan claves de 2048 bits [53]. La generación de claves lleva más tiempo cuanto mayor sea la clave, por lo que se tarda más en generar claves RSA seguras que claves de menos bits, utilizadas, por ejemplo, en criptosistemas de curvas elípticas. Este algoritmo puede ser utilizado para realizar firma digital, que puede ser utilizada, por ejemplo, para la red P2P de Gameover Zeus.

Criptosistema de curvas elípticas

El criptosistema de curvas elípticas (EC) tiene ciertas ventajas sobre el de RSA. En el caso de EC el problema matemático en el que radica su seguridad es del cálculo de logaritmos discretos de una curva elíptica. Estos cálculos son 10 veces más eficaces que los de RSA: la multiplicación de RSA pasa a ser la suma en EC, la exponenciación pasa a ser el producto. Estas operaciones son fáciles de implementar tanto en software como en hardware. Por otra parte, se ha probado que el logaritmo discreto sobre curvas elípticas es tan difícil de romper como sobre otros grupos, como los utilizados en Diffie-Hellman o ElGamal. Así pues, estas curvas se han utilizado para implementar estos algoritmos, con la misma seguridad, pero con la ventaja de utilizar claves de longitudes más pequeñas. Esto requiere menos memoria computacional y elementos hardware de tamaño más reducido [23]. Por ejemplo, en CTB-Locker se utiliza cambio de clave Diffie-Hellman para curvas elípticas. La Figura 14 presenta las longitudes de clave de EC y RSA equivalentes en fortaleza.

EC	RSA
192	1536
224	2048
384	3072
521	15360

Figura 14. Tabla de equivalencias de longitudes de clave en RSA y curvas elípticas

6.4. Funciones hash

Como se ha visto previamente (ver punto *2. Introducción*), las funciones hash son muy útiles para firma digital. Además, son útiles para generar claves únicas para cada fichero. Esto último permite cifrar archivos con cifrado simétrico utilizando como clave el hash calculado

de un archivo. Al cifrar el archivo cambia su contenido y, por tanto, cambia también su hash. Este cambio de hash evita que se pueda calcular la clave de descifrado con un solo cálculo de hash. Una vez cifrado el archivo se debe proteger la clave privada, es decir, el hash del archivo original. Para ello se emplea la criptografía de clave pública (ver punto *7. Criptografía conveniente*). El único ataque que se conoce contra algunas de estas funciones es la investigación exhaustiva. Existen tablas de equivalencias que pueden ser indexadas, por ejemplo en bases de datos SQL, y que permiten buscar, a partir de un hash, el mensaje o clave original.

En los casos de malware analizados, las funciones utilizadas son las siguientes: MD5, SHA-1 y SHA-2, esta última con longitudes de 256 y 512 bits.

MD5

Se han encontrado algunas vulnerabilidades que hacen cuestionable su seguridad. Es posible realizar ataques de fuerza bruta contra esta función buscando colisiones debido a la reducida longitud del hash, de sólo 128 bits. Es recomendable utilizar otras funciones hash más seguras.

SHA-1

La seguridad a largo plazo de SHA-1 es cuestionada debido a que existen ataques que buscan colisiones y que han reducido el espacio de posibles resultados de la función hash entre 2000. La longitud del resultado de esta función es de 160 bits.

SHA-2 y SHA-3

Estas dos funciones hash comparten una serie de cualidades como el hecho de tener una salida de tamaño variable (224, 256, 384 y 512 bits). La principal diferencia entre ellas radica en el algoritmo empleado para calcular el hash de un mensaje. La función SHA-3 está más optimizada para hardware que la función SHA-2, pero reduce su eficiencia en software.

6.5. Comparativa

Para comparar la seguridad de los algoritmos de cifrado modernos se realiza un análisis de su capacidad de resistir frente a un ataque de fuerza bruta para diferentes tamaños de clave, debido a que los problemas matemáticos utilizados son suficientemente complejos como para resistir frente a otros ataques. Algunos de los algoritmos descritos tienen ciertas vulnerabilidades ya descritas, por lo que no se incluirán en la comparativa. Siempre hay que tener en cuenta que la clave debe ser aleatoria, para poder evitar ataques de diccionario que reducen, en la mayoría de los casos, el tiempo computacional necesario a unos pocos segundos o minutos.

6.5.1. Factores determinantes

Los factores que deben tenerse en cuenta a la hora de comparar los distintos criptosistemas son [32]:

Adaptabilidad

Es la posibilidad de cambiar dinámicamente los parámetros de cifrado y los datos que es necesario cifrar, en función de ciertos requisitos o en función de la aplicación en la que se esté utilizando el criptosistema.

Velocidad de procesado

La velocidad de procesado es importante a la hora de realizar cifrado en tiempo real. Para el cifrado de archivos de mucho tamaño, por ejemplo en el caso de ramsonware, es necesario que la velocidad de procesado del criptosistema sea alta, para cifrar el máximo número posible de archivos antes de que el usuario se dé cuenta de que su máquina ha sido comprometida.

Tamaño de clave

En este apartado de la tabla se comparan las posibles longitudes de clave utilizadas por los distintos algoritmos.

Relación de cifrado

Este criterio mide la relación entre el tamaño de la parte cifrada de los datos y el tamaño total de los datos. Es necesario que esta relación sea lo más pequeña posible para reducir la complejidad de procesado.

Problemas de seguridad

Se incluyen en esta categoría los posibles ataques que puede sufrir el criptosistema:

- *Ataque man-in-the-middle*: consiste en observar e interceptar mensajes entre dos entidades que se comunican sin ser detectado. En algunos casos puede realizar cambios en los datos en tránsito.
- *CPA*: la finalidad del *ataque de texto elegido* o *chosen-plaintext attack* (*CPA*) es ganar información obteniendo el texto cifrado de un texto en claro conocido.
- *KPA*: el *ataque de texto conocido* o *known-plaintext attack* (*KPA*) se basa en el conocimiento de ambos textos: en claro y cifrado. Estos pueden ser usados para intentar encontrar la clave.
- *Bit-flipping attack*: consiste en cambiar el texto cifrado de forma que haya un cambio predecible en el texto plano.

- *Timing attack*: consiste en analizar el tiempo que tarda en ejecutarse un algoritmo criptográfico.
- *Ataque de escucha*: también llamado *eavesdropping*. Consiste en interceptar una comunicación privada en tiempo real.

6.5.2. Comparativa de criptosistemas simétricos

En la Figura 15 se reflejan las descripciones previas de los criptosistemas de clave secreta.

	Simétrico								
	DES	3DES	RC4	RC5	AES	TEA	Camellia	Blowfish	Twofish
Tamaño de clave (bits)	56	64	40 a 2048	32, 64, 128	128, 192, 256	128	128, 192, 256	32 a 448	128, 256
Velocidad	Rápido	Rápido	Rápido	Rápido	Rápido	Rápido	Rápido	Rápido	Rápido
Relación de cifrado	Alto	Moderado	Bajo	Moderado	Alto	Moderado	Moderado	Alto	Alto
Adaptabilidad	No	No	No	No	No	No	No	Sí	No
Problemas de seguridad	Fuerza bruta	Fuerza bruta, CPA, KPA	Bit-flipping attack	Timing attack, CPA, KPA	CPA, KPA	Claves equivalentes, CPA, KPA	Fuerza bruta, timing attack	Ataque de diccionario	Fuerza bruta

Figura 15. Comparativa de criptogramas simétricos

6.5.3. Comparativa de criptosistemas asimétricos

En la Figura 16 se reflejan las descripciones previas de los criptosistemas de clave pública.

	Asimétrico		
	Diffie-Hellman	RSA	EC
Tamaño de clave (bits)	Intercambio de claves	> 1024	>192
Velocidad	Rápido	Lento	Rápido
Relación de cifrado	Alto	Alto	Moderado
Adaptabilidad	Sí	Sí	Sí
Problemas de seguridad	Timing attack	Ataque de escucha	Ataque de escucha

Figura 16. Comparativa de criptogramas asimétricos

6.5.4. Comparativa de algoritmos de compresión

En la Figura 17 se muestra una comparativa de la tasa de compresión y descompresión de varios algoritmos de compresión [12].

Compressor	Ratio	Compression	Decompression
memcpy	1.000	4200 MB/s	4200 MB/s
LZ4 fast 17 (r129)	1.607	690 MB/s	2220 MB/s
LZ4 default (r129)	2.101	385 MB/s	1850 MB/s
LZO 2.06	2.108	350 MB/s	510 MB/s
QuickLZ 1.5.1.b6	2.238	320 MB/s	380 MB/s
Snappy 1.1.0	2.091	250 MB/s	960 MB/s
LZF v3.6	2.073	175 MB/s	500 MB/s
zlib 1.2.8 -1	2.730	59 MB/s	250 MB/s
LZ4 HC (r129)	2.720	22 MB/s	1830 MB/s
zlib 1.2.8 -6	3.099	18 MB/s	270 MB/s

Figura 17. Comparativa de algoritmos de compresión

7. CRIPTOGRAFÍA CONVENIENTE

Una vez analizadas las ventajas e inconvenientes de los distintos criptosistemas, es posible realizar una valoración sobre los criptosistemas de clave pública y de clave privada que es conveniente utilizar.

Los pasos más convenientes que se deben seguir en el cifrado de un archivo son: comprimir, generar clave, añadir sal o generar un *IV* y, por último, cifrar.

El primer paso permite aumentar la entropía del fichero eliminando patrones que se repitan y dificultando los ataques por texto conocido (KPA). Es importante que el algoritmo de compresión sea capaz de comprimir los archivos a un ritmo mayor o igual al de cifrado, eliminando así un posible cuello de botella en el sistema completo.

Es conveniente seguir una serie de recomendaciones en el segundo paso:

- Cifrar cada fichero con una clave distinta.
- Evitar que la clave generada para cada fichero sea pseudoaleatoria como en el caso de Neverquest.
- Utilizar claves suficientemente largas para evitar colisiones en el caso de tener que generar un alto número de claves.
- Utilizar como clave el hash del propio fichero.

La elección de la función hash apropiada para generar la clave se basa en un conjunto de variables: vulnerabilidad, longitud, tiempo de proceso en software y tiempo de proceso en hardware. Así, se pueden descartar las funciones MD5 y SHA-1, que no pertenecen ya al estándar por ser vulnerables. Las funciones SHA-2 y SHA-3 permiten utilizar las mismas longitudes de clave, por lo que esto no permite descartar ninguna de ellas. Por otra parte, SHA-3 está optimizado para hardware y tiene un rendimiento menor en software. El tiempo de procesado necesario para SHA-3 es el doble que para SHA-2 en software, mientras que en hardware SHA-3 requiere de una cuarta parte del tiempo necesario SHA-2. Si alguien quiere romper la clave SHA-3 podrá implementar un ataque de fuerza bruta en hardware reduciendo así el tiempo de procesado a una octava parte. Dado que el malware pertenece a la categoría de software, exceptuando algunas herramientas como sniffers hardware, la elección del hash apropiado es SHA-2.

La longitud apropiada de SHA-2 dependerá del tamaño de clave deseado. Considerando el uso de AES de 256 bits, que se describe a continuación, se recomienda la utilización de SHA-2 de 256 bits.

El tercer paso puede consistir en concatenar la clave con datos aleatorios, llamados sal. Esta sal es generada para cada clave, por lo que se evitará utilizar la misma combinación hash + sal para dos ficheros iguales (el hash se calcula a partir del contenido del fichero de forma

que dos ficheros iguales con distinto nombre tienen el mismo hash). El riesgo de colisión se reducirá a medida que se alargue la sal: 32 bits de sal son suficientes para reducir la probabilidad de colisión hasta hacerla despreciable. Se puede comprobar de la siguiente forma: esta longitud de sal proporciona 2^{32} posibilidades y para llegar a un 0,01% de probabilidad de colisión de clave serían necesarios al menos 429.000 ficheros iguales. Sin embargo, dada la elección de AES como criptosistema simétrico y la necesidad de utilizar un vector de inicialización en este sistema, no se recomienda el uso de sal, debido a la carga computacional que necesita su cálculo para cada fichero.

El vector de inicialización (IV) determina el punto en que se empieza a leer una clave, es decir, en el caso de que la clave sea *ABCDEF*, un vector de inicialización de 2 indicaría que la clave se empieza a leer desde la segunda posición, quedando la clave *BCDEFA*. El vector de inicialización debe ser diferente en cada archivo de un conjunto de archivos iguales. El *National Insitute of Standards and Technology* (*NIST*) recomienda el uso de un vector de inicialización de 96 bits [47]. Esto asegura que el IV será distinto para cada archivo.

El último paso es el cifrado. Para el cifrado de archivos es conveniente utilizar un criptosistema simétrico, por su eficiencia y por la utilización de claves más reducidas frente a RSA. El criptosistema simétrico más adecuado es AES, debido a que proporciona la mayor robustez posible y no se conocen ataques diferenciales exitosos contra este criptosistema. Además, AES está adoptado como estándar desde 2002, por lo que ha sido convenientemente estudiado y se ha probado su fortaleza frente a diferentes ataques. El modo de cifrado de bloques más conveniente en AES por su rendimiento es CTR, que permite una buena paralelización [48].

Una vez escogido AES, es necesario determinar la longitud de clave más apropiada. En la actualidad, las claves de 128 bits son muy robustas, pero existe riesgo de colisión de clave en el caso de generar un alto número de ellas. Los últimos avances teóricos en computación cuántica hacen temer que, si se llegasen a construir ordenadores cuánticos totalmente operativos, los problemas numéricos considerados hoy irresolubles se podrían resolver mucho más rápidamente, en un tiempo razonable, igual a la raíz cuadrada del empleado por un ordenador tradicional [23]. AES ya se construyó teniendo en cuenta esta posibilidad, de modo que permite el uso de claves de 256 bits. Además, estas claves permiten la utilización del mencionado SHA-2 de 256 bits. Por todo ello, se recomienda utilizar claves de 256 bits.

El criptosistema mencionado es recomendable utilizarlo tanto para el cifrado de archivos de un ramsonware como para el cifrado de los archivos propios del malware, implementando una rutina de descifrado en el único archivo no cifrado. Este archivo de descifrado es conveniente comprimirlo también, pero como método simple de ofuscación. Por tanto, el esquema general de cifrado de un archivo es el siguiente:

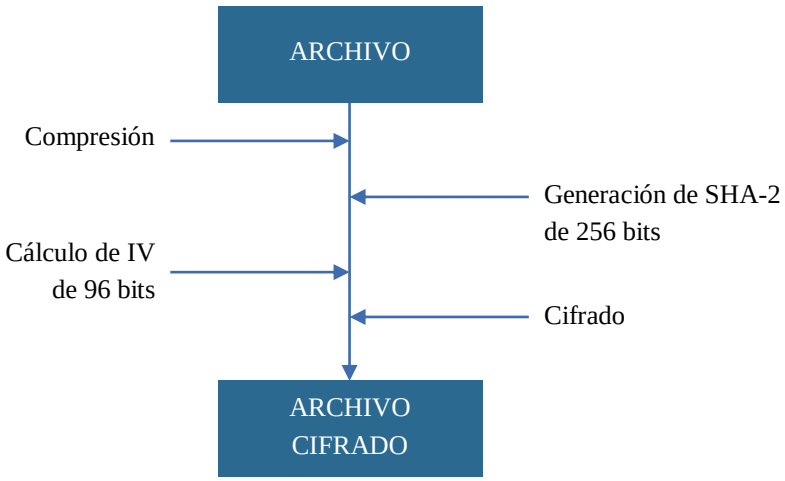

Figura 18. Proceso de cifrado conveniente

Una vez cifrados los archivos, es conveniente proteger las claves secretas utilizadas. El mejor método para protegerlas es el seguido por el ramsonware CTB-Locker, un malware que emplea unas técnicas criptográficas muy seguras. Por esta razón, los dos criptosistemas más convenientes para el manejo de las claves secretas y para realizar las comunicaciones con el panel de control son los utilizados por CTB-Locker (ver punto *5.1.1 Ramsonware, CTB-Locker*). La única diferencia entre los criptosistemas de CTB-Locker y los criptosistemas convenientes es la generación de la clave, que se ha explicado con detalle previamente en este apartado.

7.1. Carga computacional adicional

Al emplear los criptosistemas convenientes explicados en el apartado anterior, se produce en algunos casos un incremento de la carga computacional del malware correspondiente.

Como se ha indicado, el algoritmo de compresión escogido debe ser más rápido que el tiempo empleado en generar claves para un archivo y cifrarlo. Por esto, el tiempo adicional necesario para el cifrado de varios archivos será únicamente el tiempo empleado en comprimir un archivo, ya que las compresiones de los siguientes archivos se realizarán en paralelo al cifrado de los archivos ya comprimidos. En el caso de un número alto de archivos de pequeña o media longitud (menor de 3MB), el tiempo de compresión comparado con el tiempo de cifrado es despreciable.

Como se ha comentado previamente, la tasa de compresión necesaria debe ser superior a la tasa de generación del hash, explicada a continuación, que es de 112 MiB/s. El procesador empleado para calcular los datos mostrados en la Figura 17, un i5-4300U @1.9GHz, es aproximadamente cuatro veces más rápido que el empleado para medir las tasas extraídas de la referencia [11], un Core 2 1.83 GHz. En la Figura 17 se puede observar que, haciendo la conversión a MiB/s, la tasa de compresión de LZ4 es de 723 MiB/s. No se debe

confundir MiB (2^{20} bytes) con MB (10^6 bytes). Esta tasa de compresión en el procesador utilizado para los cálculos posteriores es, aproximadamente, de 180 MiB/s, que es mayor que 112 MiB/s.

La tasa de generación de SHA-2 de 256 bits es de 112 MiB/s en el estudio realizado por Crypto++ [11]. Como se comprueba a continuación, esta tasa de procesado es un 16,7% mayor que la tasa de cifrado del método AES empleado. Del mismo modo que ocurre con el proceso de compresión, la carga computacional añadida al generar el hash únicamente se debe al hash del primer archivo. Por tanto, para un número alto de archivos será despreciable.

Por otro lado, según el mismo estudio, la tasa de cifrado de AES de 256 bits en modo CTR es de 96 MiB/s, a los que hay que añadir un tiempo de establecimiento de clave y de IV de 0,756 µs. Esta tasa es mucho mayor que la tasa calculada para otros algoritmos de cifrado simétricos como XTEA (26 MiB/s + 0,636 µs) o RC5 (75MiB/s + 2,549 µs). De los algoritmos de cifrado utilizados por los malware analizados, únicamente RC4 y los algoritmos personalizados tienen una mayor tasa de cifrado. En el resto de casos, el uso de AES de 256 bits no supone una carga computacional adicional e incluso supone una mejora en algunos casos.

Algoritmo	Tasa (MiB/s)	Cálculo de IV (µs)
RC4	126	2,703
SHA-2 de 256 bits	112	
AES-256 / CTR	96	0,756
AES-256 / CBC	80	0,619
RC5	75	2,549
Twofish / CTR	59	7,716
Camellia-256 / CTR	37	0,696
DES	32	8,372
XTEA	26	0,636

Figura 19. Comparación de la tasa de cifrado de algoritmos simétricos y hash

En la Figura 20 se puede observar cómo afecta la paralelización de los procesos al tiempo total de procesado.

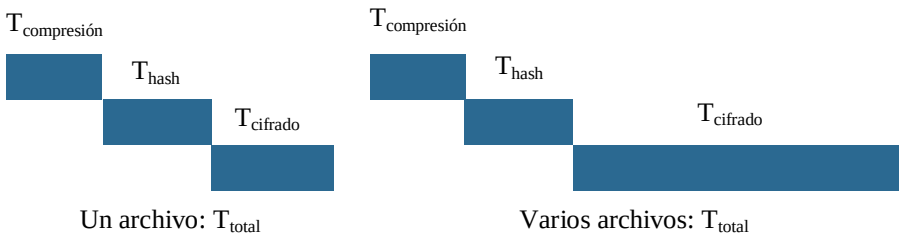

Figura 20. Paralelización del proceso de cifrado

Veamos una comparación de los tiempos requeridos para el cifrado de uno o varios archivos de 3MiB para dos casos distintos: criptosistema conveniente y criptosistema de un malware actual como TorLocker:

	Un archivo		1000 archivos	
	Criptosistema conveniente (ms)	Criptosistema de TorLocker (ms)	Criptosistema conveniente (ms)	Criptosistema de TorLocker (ms)
$T_{compresión}$	16,7	0	16,7	0
T_{hash}	26,8	0	26,8	0
T_{IV}	0	0	0	0
$T_{cifrado}$	31,25	31,25	31250	31250
T_{total}	74,75	31,25	31293,5	31250

Figura 21. Comparación de tiempos de computación

Siendo *n* el número de archivos, los cálculos realizados son los siguientes:

$T_{compresión}$ = 3 MiB / 180 MiB/s T_{hash} = 3 MiB / 112 MiB/s

T_{IV} = 0,756 µs ≈ 0 ms $T_{cifrado}$ = n * 3 MiB / 96 MiB/s

$T_{total} = T_{compresión} + T_{hash} + T_{IV} + T_{cifrado}$

Como se puede ver, la diferencia entre ambos sistemas es mínima, gracias a la paralelización de procesos.

Finalmente, el cifrado ECDH emplea diez veces menos carga computacional que la empleada por RSA (ver punto *6.3 Criptografía asimétrica*), por lo que el cambio de RSA a ECDH no añade carga computacional al sistema. Al utilizar claves de longitud más pequeña, la generación de los pares de claves aleatorias de ECDH también requiere de menos carga computacional.

Se puede concluir que el uso de los criptosistemas recomendados en este apartado aporta una pequeña carga adicional debida a la compresión y al cálculo del SHA-2 de 256 bits de los ficheros, pero, en cambio, aporta una seguridad criptográfica mucho mayor.

8. CONCLUSIONES

Los objetivos que se plantearon al comienzo de este trabajo eran múltiples y, cada uno de ellos, de gran calado. Conseguir estos objetivos ha supuesto un trabajo laborioso de investigación y análisis de múltiples fuentes.

Sin embargo, la resolución de un problema actual, como es el último objetivo, es el verdadero valor del trabajo. El criptosistema propuesto seguirá siendo seguro durante los próximos años pese al avance de los mecanismos de criptoanálisis y de los recursos disponibles. Además, este criptosistema tiene en cuenta el crecimiento continuo de la cantidad de información que se necesita asegurar siendo así escalable tanto en tamaño de archivos como en número de archivos. Esto se ha estudiado mediante el análisis de la carga computacional adicional que impone este sistema frente a otros utilizados en el ámbito del malware. Esta propuesta puede aplicarse, por otra parte, a ámbitos no relacionados con el malware como son instituciones públicas o empresas que necesitan transmitir su información de forma criptográficamente segura.

8.1. Valoración

Como valoración final, se puede decir que el trabajo ha cumplido los objetivos marcados, que realmente han sido una secuencia de objetivos para llegar a la solución concreta, práctica y que aporta valor a la visión del malware actual.

Por mi parte, he de destacar dos aspectos. En primer lugar, que me ha resultado de gran valor acometer este trabajo y profundizar en cómo se aplica la criptografía en el mundo real. En segundo lugar, que complementa mi aportación a la beca de prácticas que estoy disfrutando en la empresa S21sec, dedicada, entre otros, a este tema.

En relación con este último punto, me gustaría agradecer a esta empresa la información proporcionada, su interés y las correcciones que me han propuesto, especialmente a mi tutor del trabajo, Fernando Braquehais. Naturalmente este agradecimiento se extiende a la profesora Dra. Carmen Sánchez Ávila, por su interés y apoyo, así como por las facilidades que me ha brindado para llevar a cabo este trabajo. A ambas partes, gracias por mi primera experiencia de trabajo universitario.

8.2. Líneas de trabajo futuras

Como se ha podido apreciar, cada uno de los cinco objetivos señalados al principio del trabajo es, en realidad, un campo extenso y profundo que merece la pena ser investigado. La innovación creada a partir de la relación entre los conceptos de criptografía y malware también es un punto de partida importante para un estudio en profundidad y que parece necesario para el futuro. Los resultados de este trabajo pueden aplicarse en muy diversos campos, tanto para la seguridad de las empresas como para la seguridad de las instituciones.

9. BIBLIOGRAFÍA

[1] ÁLVAREZ HARO, J. "Estudio del algoritmo de cifrado Camellia", Universidad Politécnica de Madrid.

[2] ÁLVAREZ HARO, J. "Estudio del algoritmo de cifrado TEA", Universidad Politécnica de Madrid.

[3] ANTIY LABS "Analysis Report on Flame Worm Samples Version 1.3.0" http://goo.gl/Kd6ges

[4] ARBOR NETWORK "Beta Bot – A Code Review" https://goo.gl/4g7Di5

[5] Avast BLOG! ® "Tiny Banker Trojan targets customers of major banks worldwide" https://goo.gl/qoX0Iv

[6] AVG "Analysis of Banking Trojan Vawtrak. White Paper" http://goo.gl/OMcChB

[7] BLEEPINGCOMPUTER "TeslaCrypt and Alpha Crypt Ransomware Information Guide and FAQ" http://goo.gl/jo1hLr

[8] Blueliv "Chasing cybercrime: networks insights of dyre and dridex Trojan bankers" https://goo.gl/pGV78N

[9] CERT POLSKA "ZeuS-P2P monitoring and analysis" http://goo.gl/pKAP52

[10] Crypto++ "CTR Mode" http://goo.gl/Jhn4W7

[11] Crypto++ "Benchmarks 5.6.0" http://goo.gl/AKplth

[12] CYAN4973 "LZ4 – Extremely fast compression" https://goo.gl/Gbwb2l

[13] CYACTIVE FUTURE PROOF SECURITY "Black Friday, Cyber-Attack Monday" http://goo.gl/QvjY50

[14] DEFENCE INTELLIGENCE "Mariposa Botnet Analysis" http://goo.gl/8o6SKv

[15] DELETEMALWARE "Malware Removal Instructions" http://goo.gl/P8PSVE

[16] DELL SECURE WORKS "Top Banking Botnets of 2013" http://goo.gl/sEHqEo

[17] DELL SECURE WORKS "Dyre Banking Trojan" http://goo.gl/8R3M2J

[18] DR WEB ® ANTI-VIRUS "Adware.Mac.InstallCore.1" http://goo.gl/jvlnkF

[19] EMSISOFT BLOG "Cryptolocker, a new ramsomware variant" http://goo.gl/jZoQiB

[20] FAKEBIT "Preying on Assumptions: Symmetric Encryption in a CryptoLocker Variant" http://goo.gl/A8WMvy

[21] FIREEYE "Poison Ivy: Assessing Damage and Extracting Intelligence" https://goo.gl/VTA3ZE

[22] FORTINET "Bublik Downloader Evolution" http://goo.gl/wu3PHI

[23] FÚSTER SABATER, A. ET AL "Técnicas criptográficas de protección de datos", Editorial Ra-Ma (3ª Ed).

[24] GITHUB GIST "hzeroo/Carberp" https://goo.gl/hYV3s

[25] GITHUB GIST "RC1140 cascade.asm" https://goo.gl/camRTY

[26] GITHUB GIST "wt/coreboot" https://goo.gl/YUVSqV

[27] GÓMEZ VIEITES, Á. *Enciclopedia de la Seguridad Informática*, Editorial Ra-Ma.

[28] GUTIÉRREZ, P. "Tipos de criptografía: simétrica, asimétrica e híbrida", Genbeta Dev http://goo.gl/mVULoN

[29] INSIGHTPARTNERS® "TorrentLocker – New Variant with New Encryption Observed in the Wild" http://goo.gl/eIs9on

[30] INSTITUTO NACIONAL DE SEGURIDAD (INCIBE) "Click fraud. Descubriendo de abusos de pago por clic" https://goo.gl/9V1sYq

[31] IT CONSULTANCY SERVICES "The GP code" https://goo.gl/mZt9tP

[32] JEEVA, A.L. et al. "Comparative analysis of performance efficiency and security measures of some encryption algorithms", *International Journal of Engineering Research and Applications* (*IJERA*), 2/3, 2012, 3033-3037.

[33] KASPERSKY LAB "¿Qué es un virus de macro?" http://goo.gl/4CWrQc

[34] KASPERSKY LAB "Unveiling "Careto" - The Masked APT" http://goo.gl/xSdQXc

[35] KINDSIGHT SECURITY LABS "Cracking the Encrypted C&C Protocol of the ZeroAccess Botnet" https://goo.gl/U1WalW

[36] LABORATORY OF CRYPTOGRAPHY AND SYSTEM SECURITY "sKyWIper (a.k.a. Flame a.k.a. Flamer): A complex malware for targeted attacks", CrySyS Lab http://goo.gl/NS9W1

[37] LAVASOFT "Detecting Polymorphic Malware" http://goo.gl/C0oL3w

[38] MALWARE MUST DIE! "MMD-0036-2015 - KINS (or ZeusVM) v2.0.0.0 tookit (builder & panel source code) leaked" http://goo.gl/BcH079

[39] MCAFEE LABS "BackOff Malware Uses Encryption to Hide Its Intentions" https://goo.gl/XbCNwl

[40] MCAFEE LABS "Hesperus (Evening Star) Shines as Latest 'Banker' Trojan" https://goo.gl/q8CL62

[41] MCAFEE LABS "Potentially unwanted programs, Spyware and Adware" http://goo.gl/wh7bTG

[42] MCAFEE LABS "Ransom Cryptowall" https://goo.gl/QpPhQ9

[43] MCAFEE LABS "Threat glossary" http://goo.gl/Vum4I9

[44] MCAFEE LABS "What is a rat?" https://goo.gl/K9cKoq

[45] MICROSOFT MALWARE PROTECTION CENTER "Ramnit - The renewed bot in town" http://goo.gl/SfWue

[46] MNEMONIC "Banking Trojans- from our annual report" http://goo.gl/820SBh

[47] MORRIS DWORKIN "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST, 2007 http://goo.gl/fmTTAs

[48] OPEN WEB APPLICATION SECURITY PROJECT (OWAPS) "Trojan horse" https://goo.gl/68usMk

[49] PANDA SECURITY "Cómo eliminar Movies Toolbar / Music Toolbar" http://goo.gl/bnNFYp

[50] PANDA SECURITY "Glosario" http://goo.gl/itZVV9

[51] PANDA SECURITY "Informe semanal de Panda Security sobre virus e intrusos" http://goo.gl/yEbDLX

[52] PREVX ADVANCED MARLWARE RESEARCH TEAM "Carberp - a modular information stealing Trojan" http://goo.gl/OB2UQ

[53] REDES@ZONE "Criptografía: Algoritmos de cifrado de clave asimétrica" http://goo.gl/R7Gzve

[54] REDES@ZONE "Criptografía: Algoritmos de cifrado de clave simétrica" http://goo.gl/3p10ba

[55] RSA SPEAKING OF SECURITY "Citadel Outgrowing its Zeus Origins" https://goo.gl/WWJXcv

[56] SANS DIGITAL FORENSICS AND INCIDENT RESPONSE BLOG "TorrentLocker Unlocked" https://goo.gl/qLlVNf

[57] SECURELIST "A flawed ransomware encryptor" https://goo.gl/kw63Nz

[58] SECURELIST "A new generation of ransomware. Elliptic curve cryptography + Tor + Bitcoin" https://goo.gl/gG1qao

[59] SECURELIST "Full Analysis of Flame's Command & Control servers" https://goo.gl/weG2lk

[60] SECURELIST "ZeuS GameOver, Brazilian Trojans and Boletos: an explosive combination" https://goo.gl/KZ6TX3

[61] SCHNEIER ON SECURITY "The Blowfish Encryption Algorithm" https://goo.gl/3rl0WZ

[62] SCHNEIER ON SECURITY "Upper bounds on differential characteristics in Twofish" https://goo.gl/vgGbP8

[63] SCHNEIER ON SECURITY "Twofish" https://goo.gl/I80HfW

[64] SPIVEY, M. "Cybersecurity and its ten domains", Kennesaw University https://goo.gl/vsK2IH

[65] SPYBOT "SPAM frauds, fakes, and other malware deliveries..." https://goo.gl/I5PUAx

[66] STOP MALVERTISING MALWARE REPORTS "Analysis of Asprox and its New Encryption Scheme" http://goo.gl/aSmwvD

[67] STOP MALVERTISING MALWARE REPORTS "Analysis of Trojan:Win32/Miuref.A" http://goo.gl/v0CRU3

[68] SYMANTEC SECURITY RESPONSE "GPGPU and threat analysis" https://goo.gl/XbVEVd

[69] SYMANTEC SECURITY RESPONSE "Regin: Top-tier espionage tool enables stealthy surveillance" http://goo.gl/CdEQD3

[70] TECHREPUBLIC "10+ things you should know about rootkits" http://goo.gl/qZYC6S

[71] THOTH "Píldoras formativas en seguridad de la información" http://goo.gl/3XzDlg

[72] THREAT POST "GameOver Zeus Now Using Encryption to Bypass Detection" https://goo.gl/M36zG0

[73] TOM'S GUIDE "What is a banking Trojan?" http://goo.gl/Ioqjq5

[74] TREND MICRO INCORPORATED "Fakem Rat-Malware Disguised as Windows®, Messenger and Yahoo!® Messenger" http://goo.gl/T9gdFq

[75] TYPES LIST "Different types of computer viruses" http://goo.gl/MEXoSb

[76] VERACODE "Common Malware Types: Cybersecurity 101" https://goo.gl/gxu6Un

[77] VILLAGRÁ GONZÁLEZ, V.A. *Seguridad en Sistemas y Redes de Telecomunicación. 1. Seguridad de la información.Criptografía*. Universidad Politécnica de Madrid.

[78] VILLAGRÁ GONZÁLEZ, V. A. *Seguridad en Sistemas y Redes de Telecomunicación. 5. Riesgos de Internet*. Universidad Politécnica de Madrid.

[79] WIKIPEDIA "Luhn algorithm" https://goo.gl/sTmWN7

[80] WINDOWS IT PRO "Symmetric vs. Asymmetric Ciphers" http://goo.gl/l6UFK5

[81] WINDOWS SERVER "Remote Server Administration Tools for Windows 7" http://goo.gl/wh7bTG

[82] WINZIP "About Encryption" http://goo.gl/b2xOKL

[83] ZIARATI, R. ET AL "On the Reverse Engineering of the Citadel Botnet", Arxiv http://goo.gl/ymJooM